Metropolia

Information Technology

Multimedia Communications

MASTER'S THESIS

HIGH DEFINITION VIDEO STREAMING USING H.264 VIDEO COMPRESSION

Author: Yassine Bechqito

Supervisor: Jouko Kurki

Principal Instructor: Jouko Kurki

Approved: 15.12.2009

Jouko Kurki

Principal Lecturer

Metropolia

ABSTRACT

Name: Yassine Bechqito

Title: High Definition Video Streaming Using H.264 Video Compression

Date: 10 December 2009 Number of pages: 54

Degree Program: Information Technology

Specialization: Multimedia Communications

Instructor: Jouko Kurki, Principal Lecturer

Supervisor: Jouko Kurki, Principal Lecturer

This thesis presents high definition video streaming using H.264 codec implementation. The experiment carried out in this study was done for an offline streaming video but a model for live high definition streaming is introduced, as well.

Prior to the actual experiment, this study describes digital media streaming. Also, the different technologies involved in video streaming are covered. These include streaming architecture and a brief overview on H.264 codec as well as high definition technology. In addition, details about different streaming protocols are discussed.

The implementation of an offline high definition video experiment was carried out using Darwin Streaming Server, QuickTime Player Pro and both QuickTime and VLC players. Encoding the video using H.264 is described. The streaming inside a LAN was successful but streaming over the Internet turned out to be challenging. Possible solutions, mainly about overcoming the firewall restriction, were investigated and proposed. A streaming video hosting service was used to make streaming over the Internet possible. Finally, the streaming results are presented.

A live high definition experiment was not implemented due to the high cost of the hardware encoders available. Instead, a proposal for making the experiment possible is introduced.

Key words: Streaming, High Definition, HD, H.264, QuickTime, RTP, RTSP, Darwin Streaming Server.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I would like to thank my teacher and supervisor Mr. Jouko Kurki for offering this interesting subject for my thesis. Despite of his limited time, Mr. Kurki has been always ready to provide advice, guidance and help during the whole journey of writing this thesis.

My gratitude and respect also go out to all my teachers at Helsinki Metropolia University of Applied Sciences. Thanks to them and their great lectures, I was able to learn and discover new things that broadened my knowledge in new and exciting technologies and multimedia areas.

Last but not least, many thanks to all my family members who were always there to support and inspire me towards continuous learning and education.

LIST OF ABBREVIATIONS

AAC	Advanced Audio Coding
AVC	Advanced Video Coding
AVCHD	Advanced Video Codec High Definition
CABAC	Context Adaptive Binary Arithmetic Coding
CAVLC	Context Adaptive Variable Length Coding
CDN	Content Delivery Network
DSS	Darwin Streaming Server
DV	Digital Video
EH	Enhanced Definition
FMS	Flash Media Server
HD	High Definition
HDV	High Definition Video
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IEC	International Electrotechnical Commission
IP	Internet Protocol
ISO	International Organization for Standardization
ISP	Internet Service Provider
ITU-T	International Telecommunication Union-Telecommunication
LAN	Local Area Network
MPEG	Moving Picture Experts Group
MPLS	Multiprotocol Label Switching
NAT	Network Address Translation
PC	Personal Computer
QoS	Quality of Service
QSS	QuickTime Streaming Server
RTCP	Real-Time Control Protocol
RTMP	Real Time Messaging Protocol
RTP	Real-Time Protocol
RTSP	Real Time Streaming Protocol
SAP	Session Announcement Protocol

SD	Standard Definition
SDI	Serial Digital Interface
SDP	Session Description Protocol
SMIL	Synchronized Multimedia Integration Language
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VCD	Video CD
VCEG	Video Coding Experts Group

1. INTRODUCTION

1.1 Introduction to Streaming

Streaming is the method of transmitting media as a continuous stream of data that can be processed by the receiving computer before the entire file has been completely sent. Streaming video is a content sent in a compressed form over the Internet and displayed by the viewer in real time. With streaming video or streaming media, a Web user does not have to wait to download a file to play it. Instead, the media is sent in a continuous stream of data and is played as it arrives. The user needs a player, which is a special program that decompresses and sends video data to the display and audio data to speakers. A player can be either an integral part of a browser or specialized software.

Streaming video is usually sent from prerecorded video files, but can be also distributed as part of a live broadcast. In a live broadcast, the video signal is converted into a compressed digital signal and transmitted from a special Web server that is able to do multicast, sending the same file to multiple users at the same time.

Figure 1.1 shows a high level overview of a streaming system infrastructure.

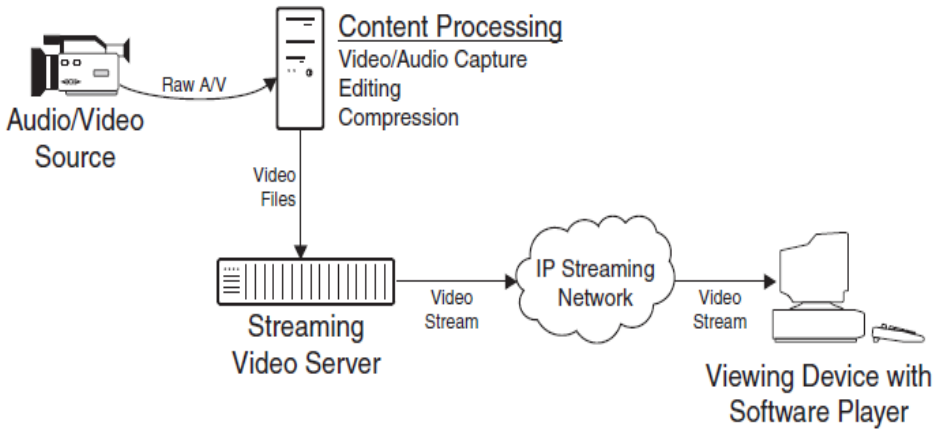


Figure 1.1: A typical streaming system infrastructure [1]

There are several methods of streaming available, the most common ones are:

- True streaming: the video signal arrives in real time and is displayed to the viewer immediately.

- Downloading: the entire file is saved on the computer, usually in a temporary folder, which then can be opened and viewed. This can be considered if, for example, the files to be viewed are small but this is not a convenient way for viewing large files, as the user needs to wait for the whole file to be downloaded before it can be viewed. Also, very large files are not playable in real time.

- Progressive downloading: the video clip is broken up into small files, each of which is downloaded to the user's device during playback, which begins playing as soon as a portion of the file has been received. This simulates true streaming, but does not have all the advantages. Progressive downloading is used mostly for delivering Flash video over the web as is the case for YouTube.

1.2 Digital Video Overview

In the past, applications for video had been limited: analog was used for broadcast and cable television, VCRs, set-top boxes, televisions and camcorders. But in recent years, there has been a huge and rapid shift to digital video, mostly based on MPEG-2 video compression standard. Today, in addition to the legacy of DV, MPEG-1 and MPEG-2 audio and video compression standards, there are three new high performance video compression standards. These new video Codecs offer much higher video compression for a given level of video quality: MPEG-4 part 2, H.264 (also known as MPEG-4 Part 10) and SMPTE VC-1 (Microsoft Windows Media Video 9 or WMV9) [2].

There are two main formats for video, which are defined by video resolution. Standard Definition (SD) is usually defined as having 576 x 720 pixels and High Definition (HD) which is usually defined as having 720p progressive or 1080i interlaced active scan lines.

High Definition Video (HDV) consumer electronics have become popular as they are available for affordable prices. These include products such as HD LCD or

Plasma television screens, camcorders, BluRay players, etc. More recently, AVCHD products have been gradually gaining ground on the market as they offer incredible HD images and more professional HD content. Advanced Video Codec High Definition (AVCHD) is a high-definition compression system. It is a highly efficient video encoder and decoder, or codec, based on MPEG-4 AVC (H.264). This is far superior to MPEG 2 encoding (used in the HDV format) [3].

Figure 1.2: Some HD Consumer products

This thesis is about High Definition Video Streaming Using H.264 Video Compression. A pre-recorded HD video will be used for streaming over the network and the tools and technology used from editing to delivery will be covered. Also, the thesis will introduce a proposal for live high definition video streaming, including techniques as well as software and hardware needed for the operation.

2. STREAMING ARCHITECTURE AND TECHNOLOGIES

2.1 Streaming Architecture

A streaming media file needs to go through several steps so that all the information can be delivered. Figure 2.1 illustrates the different stages of streaming.

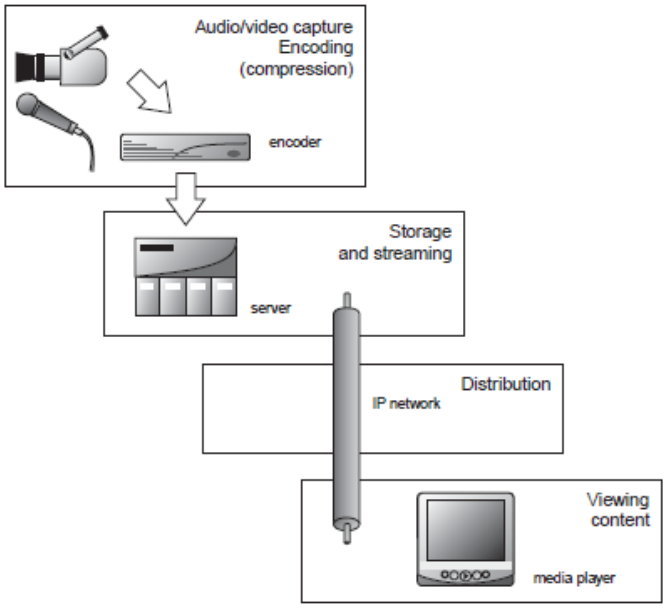


Figure 2.1: Stages of streaming [4]

The first step in streaming is to shoot the raw audio and video and then capture them to the computer file format.

The next step is to encode the captured video in a specific format such as Windows Media Streaming, QuickTime, RealNetworks Real Video, etc. Encoding is a crucial part in streaming preparation; it is here where the appropriate bitrate can be set keeping in mind whether the audience has the necessary hardware and software, and more importantly, the connection speed to support the streaming.

To deliver the encoded file to the network, it needs to be uploaded to a streaming server. Unlike a web server, the streaming server controls the stream delivery in real time, handles the load in an efficient way and increases the performance. A wide range of multimedia streaming servers are available in the market. Some of them, e.g. QuickTime Streaming Server for Apple (Darwin Streaming Server for Windows and Linux) and VLC, are free of charge. There are also some commercial ones such as Microsoft Windows Media Server from Microsoft, Adobe Flash Media Server, RealNetworks Helix Server, etc. Many protocols, as described later in this study, can be used for delivering multimedia content. The transport protocols packetize the compressed bit streams and send the video/audio packets over a LAN or to the Internet. The packets that are successfully delivered to the receiver first pass through the transport layers and

are then processed by the application layer before being decoded in the video/audio decoder.

2.1.1 Content Preparation

Due to high bitrates and large space consumption, raw video content is not suitable for streaming applications. The content needs to be processed before finally published. Figure 2.2 shows the steps involved in the streaming content preparation.

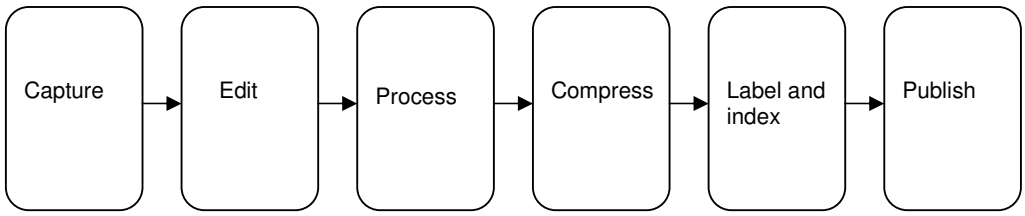


Figure 2.2: Steps of streaming content preparation

Content can come from a variety of different sources: video cameras, prerecorded tapes and DVDs, downloaded video clips, and others. In many editing systems, the content is all converted into a common format before processing takes place.

2.1.2 Streaming Server

The streaming server is responsible for distributing media streams to viewers. It takes media content that has been stored internally and creates a stream for each viewer request. These streams can be either unicast or multicast and can be controlled by a variety of mechanisms.

2.1.3 IP Streaming Network

When video is being transported over an IP network, users need to consider a number of factors such as multiplexing, traffic shaping, buffering and firewalls as

these can significantly affect the end user's viewing experience. Thus, these factors should be taken into account when planning a network.

2.1.4 Media Player

Player software that resides on the viewer's PC is responsible for accepting the incoming stream and converting it into a displayed image. Apart from just playing the media, the most intensive job of the player software is to decompress the incoming signal and create an image for display. All the player does is buffer the data packets, making sure they are in the correct order and then unpack the data packets, decompressing the digital payload [5]. The amount of processing required varies depending on the size of the image and on the compression method.

The player makes sure the data continues to stream from the source to the target client, if continuity is interrupted the player takes corrective action such as pausing, repeating frames, or rebuffering. Players may also request data to be resent.

2.2 Streaming Protocols

There are many protocols that have been developed to facilitate real-time streaming of multimedia content. Below is an overview of the most common existing protocols.

2.2.1 HTTP

The Hypertext Transfer Protocol (HTTP) is the simplest and cheapest way to stream video from a website as no dedicated streaming server is needed but only a web server that stores text and graphic files is enough for serving the HTTP streaming. Compared to, for example, RTP or RTSP protocols the latter ones always require additional tools, resources and skills for handling the streaming. This means tools such as commercial streaming server software and encoding

software and hardware or skills and resources to handle the technology used and overcome issues such as bandwidth limitations and firewall restrictions.

Small to medium-sized websites are more likely to use this method than the more expensive streaming servers.

HTTP streaming needs only a host server which recognizes common video file types and knowledge about uploading files and how to create hyperlinks.

There are some limitations to HTTP streaming: HTTP streaming is a good option for websites with modest traffic, i.e. less than about a dozen people viewing at the same time. For heavier traffic another streaming solution should be considered. This is mainly due to the streaming performance as HTTP streaming is not as efficient as other methods and will cause a heavier server load; and it can not stream live video since it only works with complete files stored on the server. Also, when using HTTP streaming, the end user's connection speed can not be automatically detected using HTTP.

2.2.2 RTP

The Real-Time Protocol (RTP) is a transport protocol that provides end-to-end network transport functions for applications transmitting data with real-time properties, such as interactive audio and video. Services that use RTP include payload type identification, sequence numbering, time stamping and delivery monitoring. The RTP packet structure details are shown in Figure 2.3.

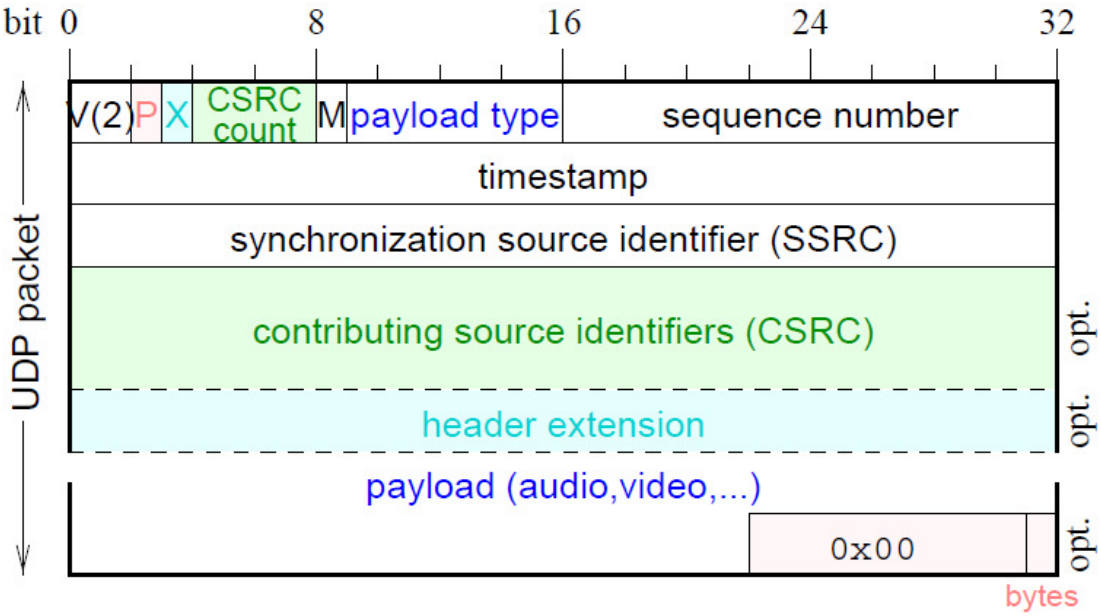


Figure 2.3: RTP packet header [6]

A brief explanation for different RTP header fields is introduced in *Appendix A*.

The most important thing RTP does is time stamping that allows placing the incoming audio and video packets in the correct timing order. Applications run RTP on top of the User Datagram Protocol (UDP). RTP includes RTCP, a closely linked protocol, to provide a mechanism for reporting feedback on the transmitted real-time data. RTP can be used in the following scenarios: multicast audio conferencing as well as audio and video conferencing.

The protocol has been demonstrated to scale from point-to-point use to multicast sessions with thousands of users, and from low-bandwidth cellular telephony applications to the delivery of uncompressed High-Definition Television (HDTV) signals at gigabit rates [6].

2.2.3 RTCP

The Real-Time Control Protocol (RTCP) is a data transport protocol used in conjunction with RTP for transporting real-time media streams. It includes functions to support synchronization between different media types (e.g., audio and video) and to provide information to streaming applications about network quality, number of viewers, identity of viewers, etc.

RTCP gives feedback to each participant in an RTP session. This feedback can be used to control performance. The messages include reception reports, including number of packets lost and jitter statistics (early or late arrivals). This information can potentially be used by higher layer applications to modify the transmission. Some RTCP messages relate to control of a video conference with multiple participants [4].

2.2.4 RTSP

The Real-Time Streaming Protocol (RTSP) is an application-level protocol for the control of real-time multimedia data. It provides a means for users to control video, audio, and multimedia sessions (e.g. *play*, *pause* and *stop*). RTSP does not actually provide the delivery of the video signals; it allows these signals to be controlled by a user. Like a dispatcher for a delivery service, RTSP does not go out and actually deliver packages; it controls when and how packages are delivered by other protocols such as RTP. Basically, RTSP acts as a remote control for the media server.

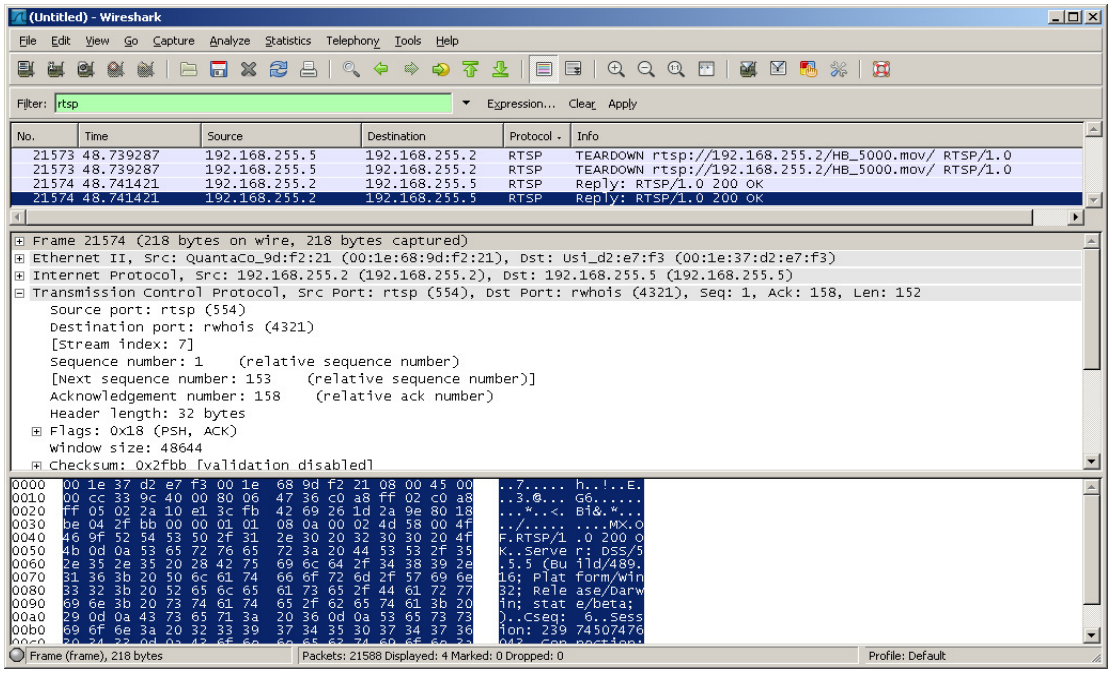


Figure 2.4: A capture screen from an RTSP stream using Wireshark

2.2.5 RTMP

Real Time Messaging Protocol (RTMP) is a proprietary streaming protocol developed by Adobe systems for streaming audio, video and data over the Internet.

RTMP uses TCP/IP protocol for streaming and data services. In a typical scenario, a web server delivers the stream over HTTP. The client creates a socket connection to Flash Media Server over RTMP. The connection allows data to stream between client and server in real time [7].

The server and the client send RTMP messages over the network to communicate with each other. The messages could include audio, video, data, or any other messages. The RTMP message has two parts: a message header, which contains message type, length, timestamp and message stream Id, and the message payload, which is the actual data such as audio samples or compressed video data that is contained in the message.

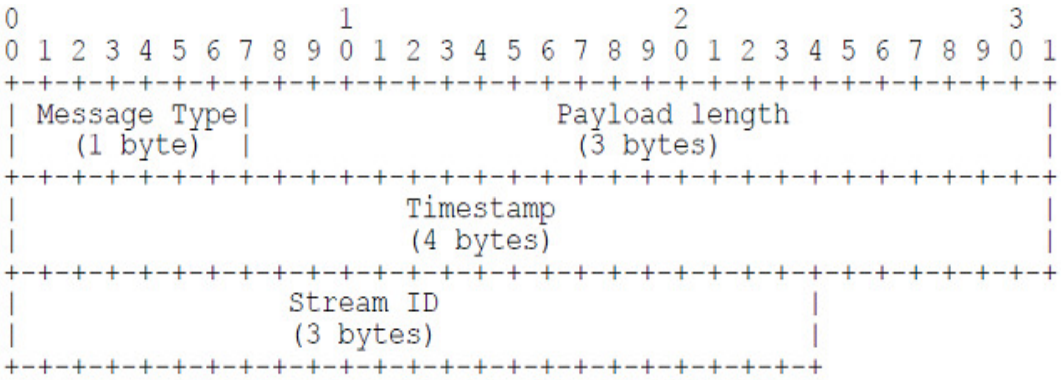


Figure 2.5: RTMP message header [8]

To demonstrate video streaming using RTMP protocol, an experiment using Adobe Flash CS3 Video Encoder, Flash Media Server 3.0 (FMS) and a player supporting RTMP streams, was implemented.

First a short HD video was encoded using Adobe Flash CS3 Video Encoder, the instructions for this step are covered in *Appendix B*. Once the video had been encoded, it was stored on the media directory of FMS (C:\Program

Files\Adobe\Flash Media Server 3\applications\vod\media). Before playing the RTMP stream on a client machine, the FMS needs to be started.

Figure 2.6: Starting Flash Media Server 3.0

The stream can be then played from a remote machine by any RTMP or Flash compatible player, the streaming address has the following format: rtmp://192.168.255.3/vod/HB_5000

Note: HB_5000 refers to the flash file name without the .flv extension; the stream will not work if the file extension is included.

Figure 2.7: Playing the RTMP stream from a Flash player

2.2.6 SMIL

The Synchronized Multimedia Integration Language (SMIL) was developed to allow the design of websites that combined many different types of media

including audio, video, text, and still images. With SMIL, the web page author can control the timing of when objects appear or play and can make the behavior of objects depend on the behavior of other objects. SMIL is a recommended XML markup language approved by the World Wide Web Consortium (W3C), and it uses ".smil" as a file extension. SMIL is supported by QuickTime, Real, and Windows Media architectures [4].

SMIL is similar to HTML and can be created using a text-based editor. SMIL scripts can be written and embedded into standard web pages to cause actions or reactions to user inputs. The language has parameters that can define the location and sequence of displays in a sequential fashion and prescribe the content layout, i.e. windows for text, video, and graphics.

SMIL can be used as an interactive presentation layer in a multimedia streaming environment. This means that the player will not only show the video stream at the client's end but, for example, text and audio files can be streamed simultaneously as a presentation. The following sample code demonstrates the usage of SMIL in a streaming multimedia presentation.

```
<smil>
<head><layout>
<root-layout height="184" width="376" background-color="blue"/>
<region id="video" left="0" top="0" height="144" width="176">
<region id="audio" left="176" top="0" height="144" width="200">
<region id="text" left="0" top="144" height="40" width="576">
</layout></head>
<body><par>
<video id="the_video" src="rtsp://192.168.255.3/HB_5000.mov" region="video"/>
<text id="the_audio" src="Nostalgie.mp3" region="audio"/>
<text id="the_text" src="textfile.txt" region="text"/>
</par></body>
</smil>
```

Figure 2.8: Sample SMIL code for playing a streaming presentation

Another example, in Figure 2.9, shows the usage of SMIL along with RTMP streaming. It checks the user's own connection speed and picks up the streaming file with the most appropriate bitrate.

```
<smil>
<head>
<meta base="rtmp://localhost/vod/" />
</head>
```

```
<body>
<switch>
<video src="world_5000.flv" system-bitrate="5000000"/>
<video src="world_3000.flv" system-bitrate="300000"/>
<video src="world_1000.flv" system-bitrate="100000"/>
<ref      src="world_500.flv" />
</switch>
</body>
</smil>
```

Figure 2.9: Sample SMIL code for picking the suitable stream bitrate

The third line in the code above refers to the RTMP streaming URL including the media directory in Flash Media Server. The other lines in the body section list the available media flash files for the playback. The files have similar content except that they are encoded at different bitrates. The last video option is the default one to be played when none of the bitrates listed in the code is detected at the client side.

2.2.7 SDP

The Session Description Protocol (SDP) describes the general real-time multimedia session. A multimedia server uses SDP to announce a conference session by periodically multicasting an announcement packet to a familiar multicast address and port using the Session Announcement Protocol (SAP).

SDP transports information about media streams in multimedia sessions and allows the recipients of a session description to participate in the session. SDP is normally used in an Internetwork but can also be used for multimedia conferences in other network environments.

For example, when using QuickTime Broadcaster and QuickTime Streaming Server (QSS) to broadcast a live video, an SDP file will be generated on the server's side. To view the live video on a remote computer, the URL including the SDP file name needs to be opened in QuickTime Player. The created SDP file is a short description file in text format that holds information about the name and the purpose of the session, protocols, codec format, media, timing and transport information. To demonstrate the usage of the SDP file in action, a small experiment was carried out using QSS on Mac OS X 10.5 along with QuickTime

Broadcaster for live streaming. The pictures below show how SDP is practically used. Detailed instructions about setting up and configuring QSS for live broadcasting are explained in Joann Karas's graduate study [9].

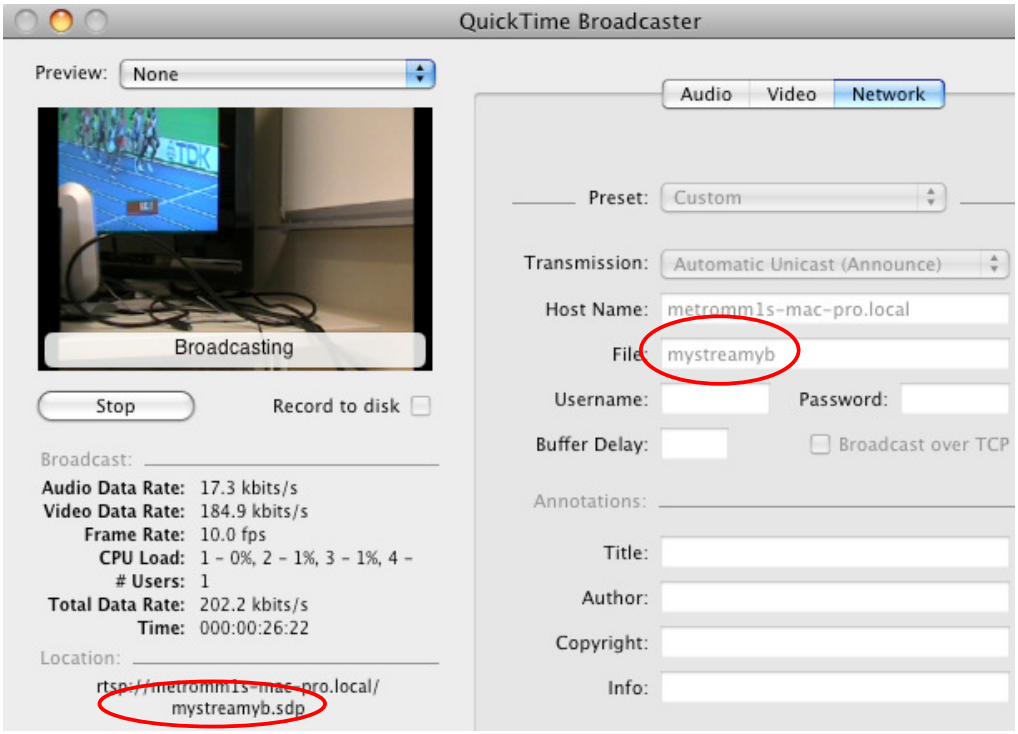


Figure 2.10: Defining the SDP file in QuickTime Broadcaster

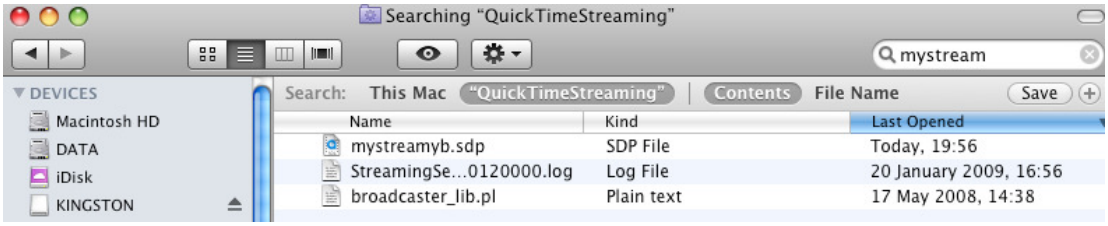


Figure 2.11: The SDP file generated on Apple QSS server

```
v=0
o=- 0 4033660108 IN IP4 127.0.0.0
s=QuickTime
c=IN IP4 127.0.0.1
t=0 0
a=range:npt=now-
m=audio 6970 RTP/AVP 96
b=AS:16
a=rtpmap:96 mpeg4-generic/8000/1
```

```
a=fmtp:96 profile-level-id=15;mode=AAC-
hbr;sizelength=13;indexlength=3;indexdeltalength=3;config=1588
a=mpeg4-esid:101
a=control:trackid=1
m=video 6970 RTP/AVP 97
b=AS:200
a=rtpmap:97 MP4V-ES/90000
a=fmtp:97 profile-level-
id=1;config=000001B0F3000001B50EE040C0CF000001000000012000844OFA285020F0A31F
a=mpeg4-esid:201
a=cliprect:0,0,240,320
a=framesize:97 320-240
a=control:trackid=2
```

Figure 2.12: The generated SDP file content

The format of the SDP file has entries in the form of <type>= <value>, where the <type> defines a unique session parameter and the <value> provides a specific value for that parameter. The session description parameters are explained in *Appendix C*.

To play the stream using SDP a Multimedia player such as QuickTime Player or VLC can be used.

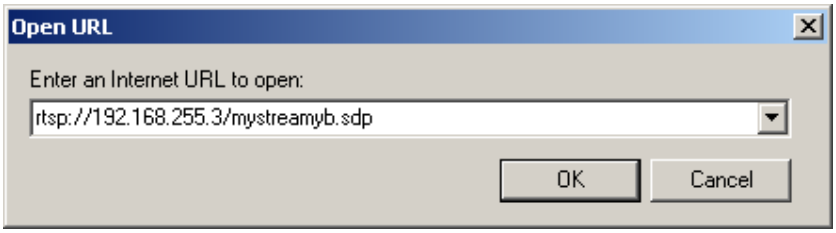


Figure 2.13: Accessing the live broadcast from QuickTime Player on a client

The URL in this contest refers to the RTSP stream address.

2.3 Streaming Media Distribution

There are three common techniques for streaming real-time audio and video over a network: unicasting, multicasting and broadcasting.

2.3.1 Unicast

A unicast stream is a one-to-one connection between the server and a client, which means that each client receives a distinct stream and only those clients that request the stream receive it. In other words, in unicasting each video stream is sent to exactly one recipient. If multiple recipients want the same video, the source must create a separate unicast stream for each recipient. These streams then flow all the way from the source to each destination over the IP network. Figure 2.14 shows how data flows under unicasting.

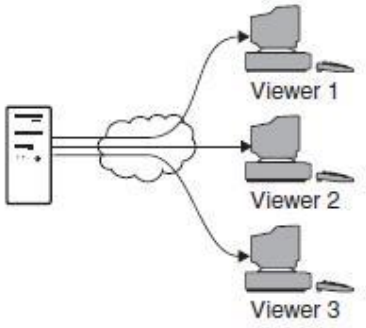


Figure 2.14: IP Unicasting [1]

2.3.2 Multicast

In multicasting, a single video stream is delivered simultaneously to multiple users. Through the use of special protocols, the network is directed to make copies of the video stream for every recipient. This process of copying occurs inside the network rather than at the video source. Copies are made at each point in the network only where they are needed. The full range of IPv4 multicast addresses is from 224.0.0.0 to 239.255.255.255.

Figure 2.15 shows how data flows under multicasting.

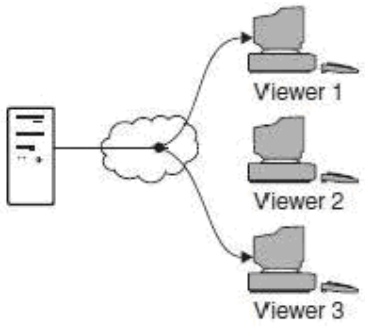


Figure 2.15: IP Multicasting

2.3.3 Broadcast

In broadcasting a single packet is sent to every device on the local network. Each device that receives a broadcast packet must process the packet in case there is a message for the device. Broadcast packets should not be used for streaming media, since even a small stream could flood every device on the local network with packets that are not of interest to the device. Broadcast packets are usually not propagated by routers from one local network to another, making them undesirable for streaming applications. In true IP multicasting, the packets are sent only to the devices that specifically request to receive them, by joining the multicast.

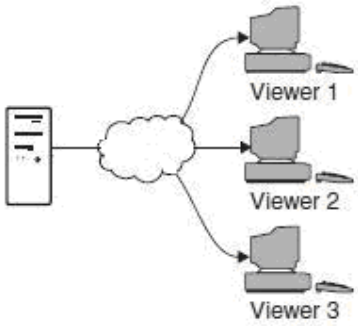


Figure 2.16: Broadcasting

2.4 Streaming Products and Formats

To stream media files in real-time, they must be wrapped by one of the streaming formats. These formats have timing control information that can be used by the

server to manage the flow rate. If the client is using interactive control, the file index aids the navigation [5].

The main file formats for streaming video files are shown in Table 2.1.

Table 2.1: Common file extensions for streaming video files

Extension	File Description	Common codecs supported
.3g2	3GPP2 Multimedia File	H.263, MPEG-4 Part 2, H.264
.3gp	3GPP Multimedia File	H.263, MPEG-4 Part 2 and H.264
.flv	Flash Video File	On2 VP6, Sorenson Spark, H.264
.mov	Apple QuickTime Movie	H.264
.mp4	MPEG-4 Video File	MPEG-4, H.264, VC-1
.mpg	MPEG Video File	MPEG-1, MPEG-2
.qt	Apple QuickTime Movie	H.264
.rm	Real Media File	RealVideo
.swf	Macromedia Flash Movie	On2 VP6, Sorenson Spark, H.264
.vob	DVD Video Object File	MPEG-2 Part 1, MPEG-2 Part 2
.wmv	Windows Media Video File	VC-1

There are many streaming video formats and products to choose from when creating video streams.

2.4.1 Microsoft Windows Media Format

Windows Media is the Microsoft family of coders and decoders as well as streaming server and players. It has the following components: Windows Media Encoder Series, Windows Media Services (server) and Windows Media Audio and Video Codecs

The encoders can take video files stored in various formats such as .avi and generate files in the .wmv or .asf format. Windows media players are available as a part of the Windows operating system. The servers for streaming are Windows Media servers and stream files in the .wmv or .asf format

2.4.2 RealNetworks

RealNetworks products offer a number of streaming-related products, including the following:

- RealPlayer, a version of player software for content encoded in RealAudio and RealVideo formats.

- Helix servers, which come in a variety of models to handle streaming functions for servers ranging from small, private streaming sites to large, professional sites that provide massive amounts of public content.

- RealProducer and Helix Producer, which take video and audio content and convert it into RealAudio and RealVideo formats for storage and playback to streaming clients.

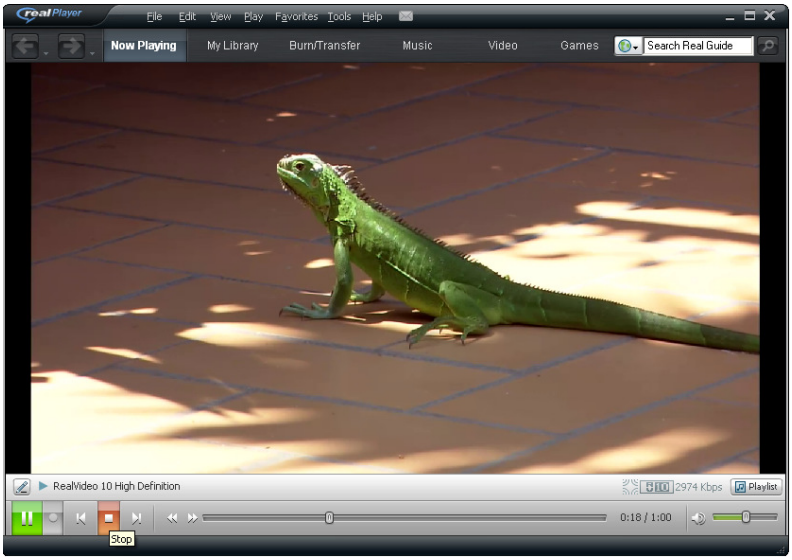


Figure 2.17: A High Definition video played using RealVideo 10 codec

2.4.3 Apple

Apple's QuickTime is a complete set of tools and players for the handling of multimedia and streaming. QuickTime components include a browser plug-in or QuickTime multimedia player and QuickTime streaming server. Apple's QuickTime format is capable of streaming the latest industry standard H.264 MPEG-4 format which offers high compression with good quality broadcast video. Apple also provides free movie-editing software (iMovie) as part of some software releases and sells a professional tool for editing movies called Final Cut Pro.

2.4.4 Adobe

Adobe's Flash platform has been widely adopted for Internet video applications, including major and popular portals. More recently, Adobe has also been offering H.264 for high quality and HD applications.
Adobe Flash Media Server (FMS) product provides streaming media capabilities and a scripting engine that enables the creation and delivery of a wide range of interactive media applications [10]. FMS is used in various applications among which Video on Demand and live video communications. It uses Real Time Messaging Protocol (RTMP), which is Adobe proprietary protocol, for streaming.

2.4.5 VLC

VLC, initially known as VideoLAN Client, is a highly portable open source media player and a powerful media server. It is available for the most common platforms such as Windows, Linux and Mac OS X.

When used as a client, VLC can play a wide range of audio and video types such as MPEG-2, MPEG-4, H.264, MP3, DivX, VCD, DVD, etc [11]. As a server, VLC supports different types of streams: UDP/RTP Unicast, UDP/RTP Multicast, HTTP, RTSP, MMS, etc [11].

Figure 2.18: VLC main user interface

Table 2.2 summarizes the difference between the most common multimedia streaming servers.

Table 2.2: Comparison between different streaming servers

	Helix Server	Windows Media Server	Flash Media Server	Darwin Streaming Server	QuickTime Streaming Server	VLC
Provider	RealNetworks	Microsoft	Adobe	Apple	Apple	VideoLAN
File types	rm, ra	wma, wmv, asf, wsx	flv, fla, swf	mov, avi, 3gpp,3gpp2	mov, avi, 3gpp, 3gpp2	Almost all file types
CODEC	RealVideo	VC-1	On2 VP6, Sorenson Spark, H.264	H.264	H.264	Multiple
Protocol	RTSP	HTTP, MMS, RTSP	RTMP (proprietary)	RTP,RTCP, RTSP	RTP, RTCP, RTSP	Multiple
Operating System	Windows, Linux, Solaris	Windows	Windows, Linux	Windows, Linux, Mac OS	Mac OS	Windows, Linux, Mac OS
Announce file type	RAM	ASX	SDP	SDP	SDP	SDP
Cost (€)	2400-14500	~700	~700	Free	Free (included in MAC OS)	Free

3. VIDEO CODECS AND HIGH DEFINITION VIDEO OVERVIEW

3.1 Codecs Overview

Streaming video and audio signals over an IP network need to be compressed in most cases, which means reducing the amount of bits that need to be transported while keeping the quality as good as possible. Compression is the process in which the amount of data used to send the video and audio is reduced to meet the bitrate requirements.

In general, a codec is a software that encodes and decodes (compress and decompress) the video streams.

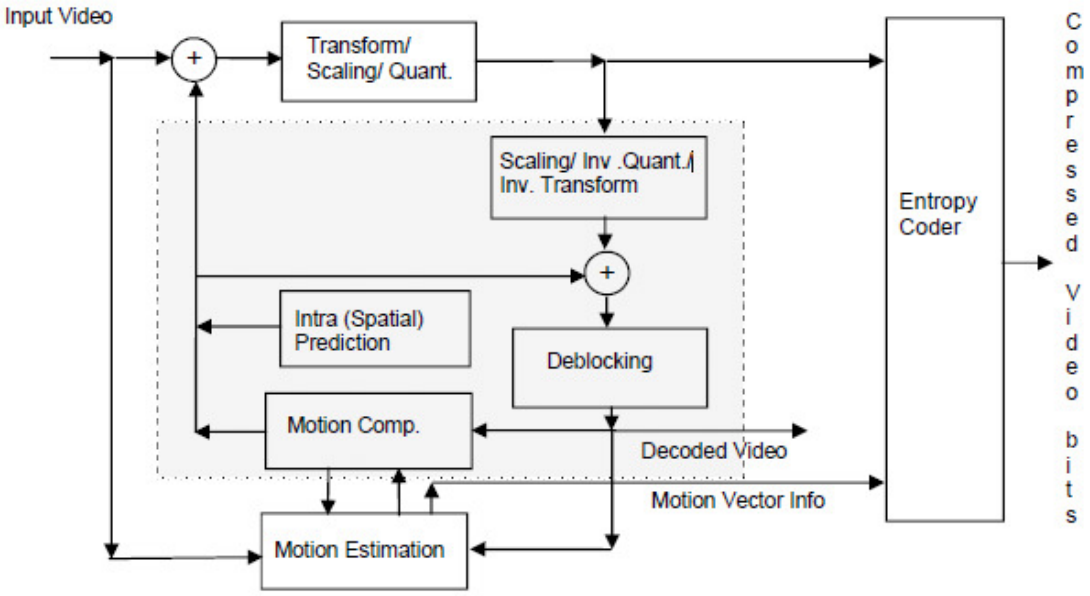


Figure 3.1: High level encoder architecture [12]

The Moving Pictures Experts Group has developed some of the most common compression systems for video around the world and given these standards the common names MPEG-1, MPEG-2, and MPEG-4.

3.2 Video Compression using H.264 Codec

H.264/MPEG4-AVC is the video coding standard of ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). H.264 has been adopted by the Motion Picture Experts Group (MPEG) to be a key video compression scheme in the MPEG-4 format for digital media exchange. It is also known as MPEG-4 Part 10 and MPEG-4 AVC (Advanced Video Coding). H.264 delivers the same quality as MPEG-2 at a third to half the data rate, and when compared to MPEG-4 Part 2, H.264 provides up to four times the frame size at a given data rate [13].

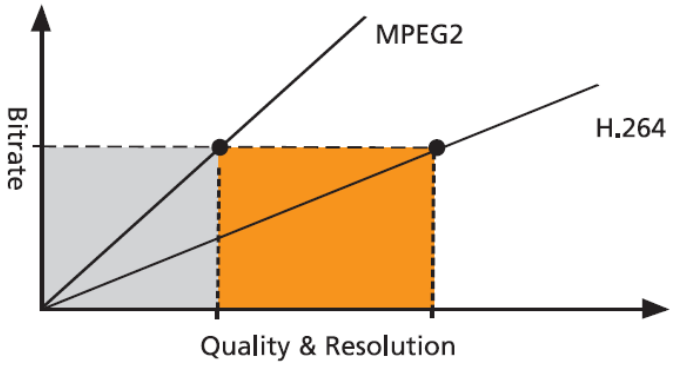


Figure 3.2: H.264 Vs MPEG2 [13]

A profile defines specific encoding techniques that you can or can not utilize when encoding the files (such as B frames), while the level defines details such as the maximum resolutions and data rates.

There are a few types of H.264, referred to as profiles [14]:

- Baseline: Intended for low-complexity applications such as video conferencing and mobile multimedia.
- Main: Intended for the majority of general uses, such as the Internet, mobile multimedia, and stored content.
- Extended: Intended for streaming applications, where stream switching technologies can be beneficial.
- Three High profiles (also known as Fidelity Range Extension or FRExt). Consists of three separate High profiles (High, High 10, and High 4:2:2), intended for high end professional uses.

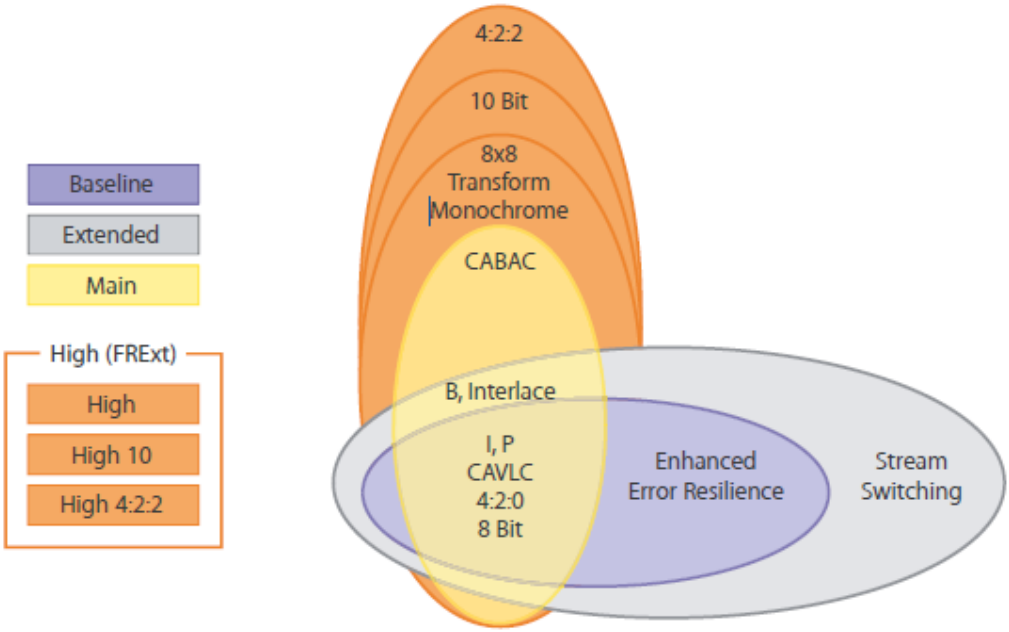


Figure 3.3: H.264 profiles [14]

For each profile, 16 levels can be applied, each specifying a typical frame size, frame rate, and maximum data rate. The same 16 levels are used for each profile. In other words, levels are used to limit performance, bandwidth and memory requirements. Each level defines the bitrate and the encoding rate in macro block per second for resolutions ranging from QCIF to HDTV and beyond. The higher the resolution, the higher the level required. H.264 is applied for many applications and consumer electronics, for example HDTV, BluRay players, etc. Table 3.1 shows the levels details.

Table 3.1: H.264 levels description [14]

Level	Typical resolution	Typical frame rate	Maximum bitrate
1	128×96 or 176×144	30 or 15	64 Kbps
1b	128×96 or 176×144	30 or 15	128 Kbps
1.1	176×144 or 320×240	30 or 10	192 Kbps
1.2	176×144 or 320×240	60 or 20	284 Kbps
1.3	352×288	30	768 Kbps
2	352×288	30	2 Mbps
2.1	352×288	50	4 Mbps
2.2	352×288 or 640×480	50 or 15	4 Mbps
3	720×480 or 720×576	30 or 25	10 Mbps
3.1	1280×720	30	14 Mbps
3.2	1280×720	60	20 Mbps
4	1920×1080	30	20 Mbps
4.1	1920×1080	30	50 Mbps
4.2	1920×1080	60	50 Mbps
5	2560×1920	30	135 Mbps
5.1	4096×2048	30	240 Mbps

A number of tools for encoding video files using H.264 codec are available; the most used ones are Apple QuickTime, Nero Recode 2 and x264 which is an open source. A description of how to use Apple QuickTime Pro for encoding the video to be streamed using H.264 will follow later in the experiment chapter.

3.3 High Definition Overview

High definition video offers much more detail in video images because it uses many more pixels than standard video. This allows much larger video displays to be used without the loss of sharpness and high quality.

HD signals use a different aspect ratio for the video image. For SD signals the aspect ratio is 4:3 (in Finland 16:9), meaning that the video image is 4 units wide by 3 units high. As per HD signals, the aspect ratio is 16:9, which is one-third wider than 4:3.

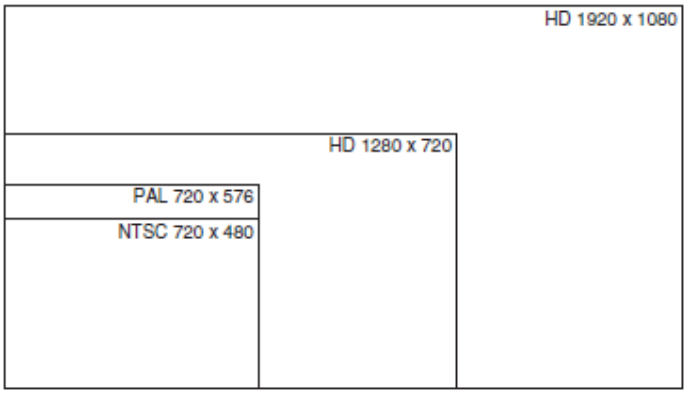


Figure 3.4: HD and SD formats comparison

There are two common forms of high definition video known as 1080i and 720p, which have several variations based on the frame rate of the video stream. Some options for each include 25 fps (frames per second), 60 fps, or 24 fps is also available and is the same frame rate used for film production. However, and due to the low frame rate, 24 fps does not work well with images involving fast motion. Table 3.2 shows the most common types of HD.

Table 3.2: Common types of HD

Type	Size	Frames per second
720 24p	1280 x 720	23.976 fps progressive
720 25p	1280 x 720	25 fps progressive
720 30p	1280 x 720	29.97 fps progressive
720 50p	1280 x 720	50 fps progressive
720 60p	1280 x 720	59.94 fps progressive
1080 24p	1920 x 1080	23.976 fps progressive
1080 25p	1920 x 1080	25 fps progressive
1080 30p	1920 x 1080	29.97 fps progressive
1080 50i	1920 x 1080	50 fields persecond/25 fps interlaced
1080 60i	1920 x 1080	59.94 fields persecond/29.97 fps interlaced

Most HDV cameras available in the market today, such as Canon and Sony, support two main resolutions: 1440 x 1080 50i for commercial use and 1280 x 720 50p for professionals.

4. HD VIDEO STREAMING EXPERIMENT

4.1 Streaming Environment Setup

4.1.1 Darwin Streaming Server Setup

Darwin Streaming Server (DSS) is free and open source server software provided by Apple, it serves as a streaming server which is based on Apple QuickTime Streaming server. While the latter one is for Mac OS X platform only, Darwin Streaming Server supports the most popular operating systems such as Linux and Windows. DSS can be used for streaming video, music, creating an Internet radio and for live broadcasting when used along with broadcaster software [15]. It also supports unicast and multicast streaming and uses RTP and RTSP protocols for streaming. A typical setting environment for DSS is depicted in figure 4.1.

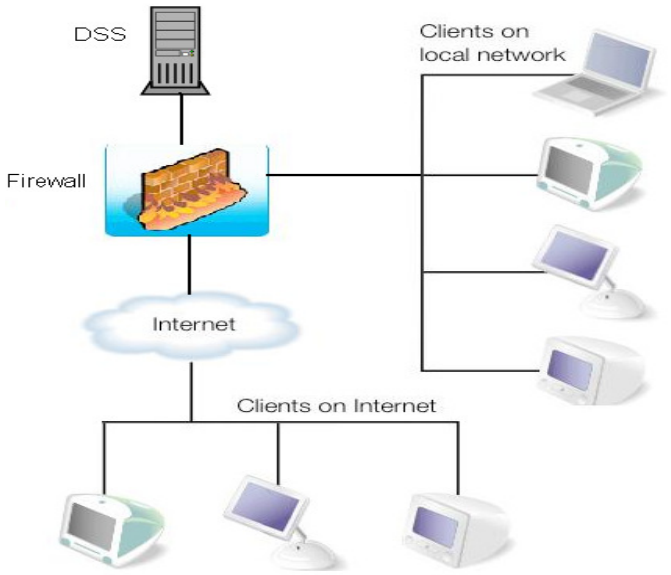


Figure 4.1: Darwin Streaming Server environment

To install DSS on Windows, follow the detailed installation and configuration steps in *Appendix D*. Once the installation is finished successfully, open the browser, type the address *http://localhost:1220* and then log in using the user name and password created during the DSS installation phase, as shown in Figure 4.2.

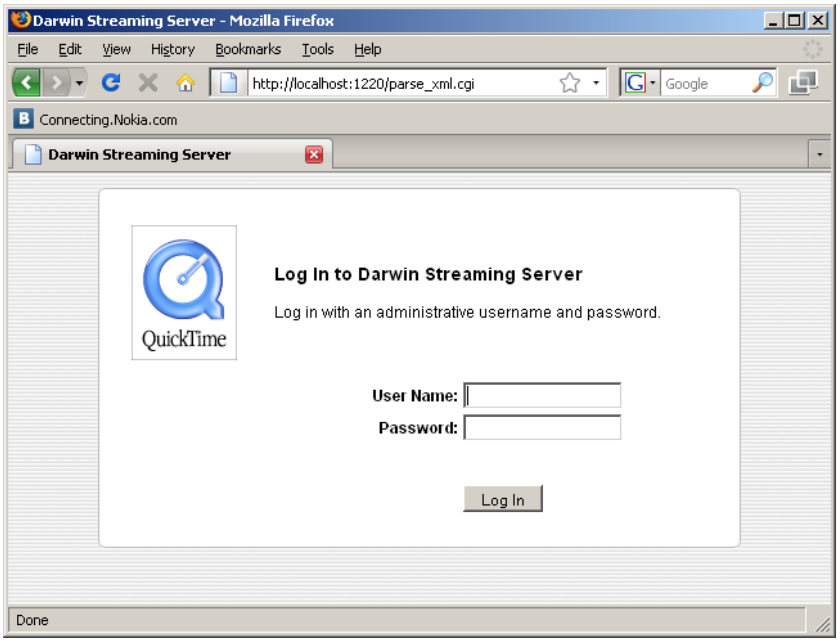


Figure 4.2: DSS administrator login page

Once the credentials have been entered, the main administrator page for DSS is open, confirming that the server is running. See Figure 4.3.

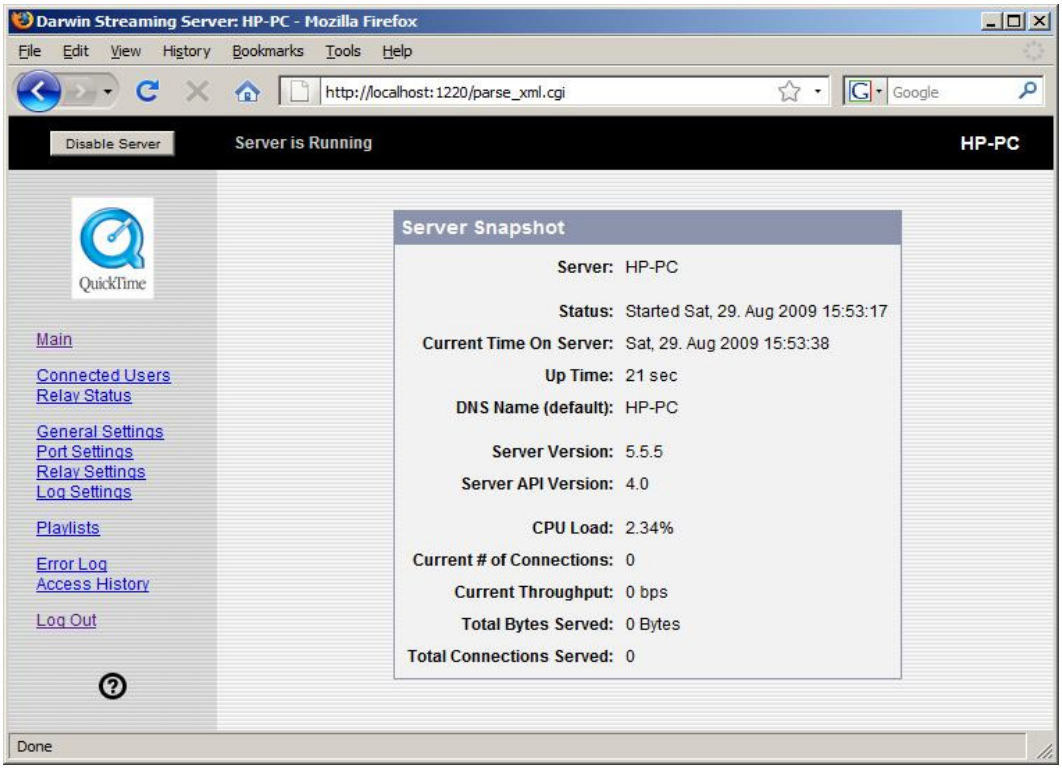


Figure 4.3: DSS main administrator page

Note: make sure to run DSS from the command line before trying to access the server and leave the DOS prompt window open, otherwise DSS will not work. Assuming that Active Perl is installed on *C:\Perl folder*, the command is:

C:\Perl\bin\perl "C:\Program Files\Darwin StreamingServer\streamingadminserver.pl"

You can create a bat file for the above command and put it in Windows startup folder so that DSS will start automatically after every Windows startup.

Figure 4.4: Sample batch file for starting up DSS automatically

You can also get some useful commands help, if needed, by typing the command shown in Figure 4.5.

Figure 4.5: Some DSS useful commands

4.1.2 Testing the Streaming Setup

Now the DSS is installed and ready for testing. To test the streaming, you can use the sample movies included in DSS default media folder. We use here QuickTime player to test but other players, such as VLC, can also be used for testing the streams.

Open QuickTime Player, go to *File* and then *Open URL*:

Figure 4.6: Opening the URL in QuickTime Player

Type the RTSP URL as shown in Figure 4.7. Here we use the sample video *sample_h264_1mbit.mp4* found in the DSS media folder.

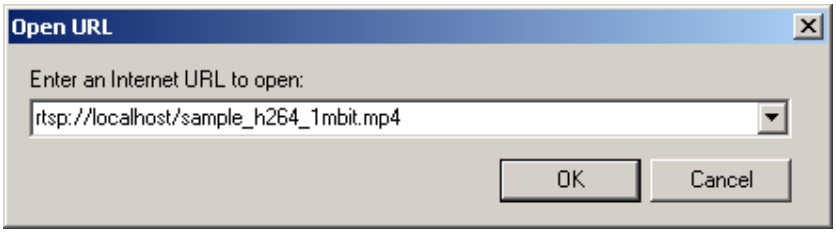


Figure 4.7: The URL for playing the sample video

Click *OK* to play the sample movie. QuickTime Player will play will connect to the streaming server and play the video.

From DSS administrator page, you can see the connected users by clicking *Connected Users* link on the right menu.

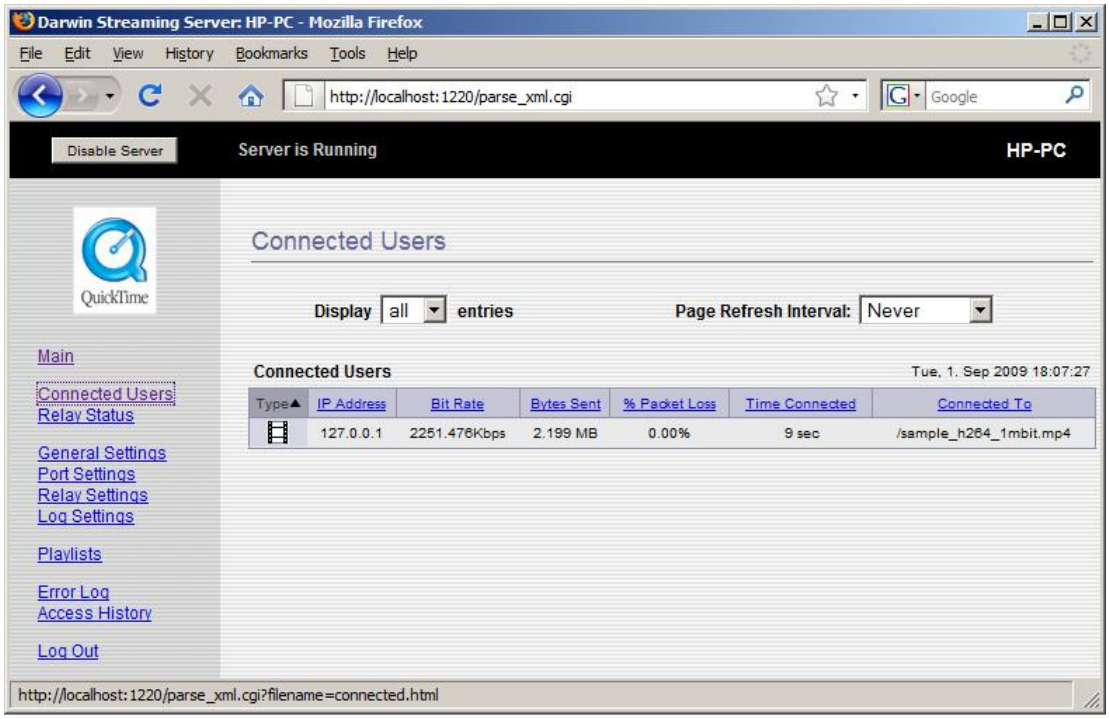


Figure 4.8: A connected remote IP as shown on DSS administrator page

Now that the streaming is working fine on the streaming server, we can go further and test by playing the video on a remote computer. This can be done by opening QuickTime Player on the remote computer and entering the RTSP URL, this time by typing the IP address of the streaming server and the movie name from the ones existing in the DSS media folder. In this experiment, the RTSP address as shown in Figure 4.9 was used.

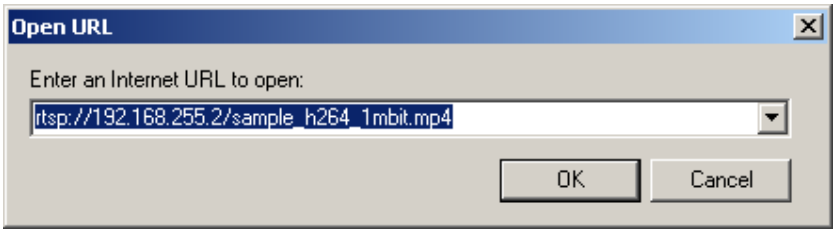


Figure 4.9: Playing a video in QuickTime Player on a remote computer

After clicking *OK*, QuickTime Player will play the video from DSS server.

It may take some time before you connect for the first time to an IP address; this means that the player checks the bandwidth to the server. Also, if there is a

firewall or the default UDP port is unavailable the client will try alternate ports and protocols to connect to the server.

If there are issues with connecting, you may need to disable your firewall on the DSS server (though this is not recommended) or add the ports to the firewall exceptions. The best solution is to configure the firewall to allow streaming access minimally on port 554 and preferably with UDP support on 6970-6999, and 1220 for web admin access, 8000 for mp3 streaming, and 7070 for some streaming players. When that is not possible, the QuickTime Player will automatically try to switch to the HTTP protocol to stream from the server. This sometimes works but if the standard streaming ports are completely blocked by a firewall then streaming on port 80 should be tried.

Streaming setup architecture using HTTP or RTSP is shown if Figure 4.10.

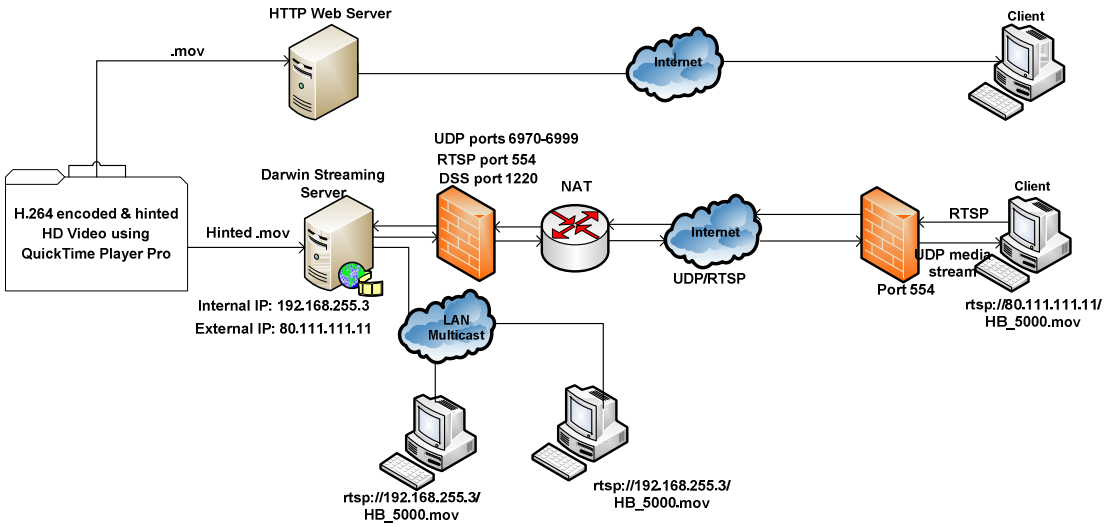


Figure 4.10: Video streaming setup using either HTTP or RTSP protocol

4.2 Streaming a high definition video

Now that the streaming environment is set up and tested successfully for streaming and playing the sample movies that come with DSS, it is time to make a true streaming experiment using a high definition video.

A video of about 100 seconds long was shot with Sony HVR-A1E HDV-camera, at 1440x1080 pixel resolution, at 25 fps frame rate. Figure 4.11 shows the video details using the Media Information tool from VLC Player (from *Tools* menu – *Media Information*).

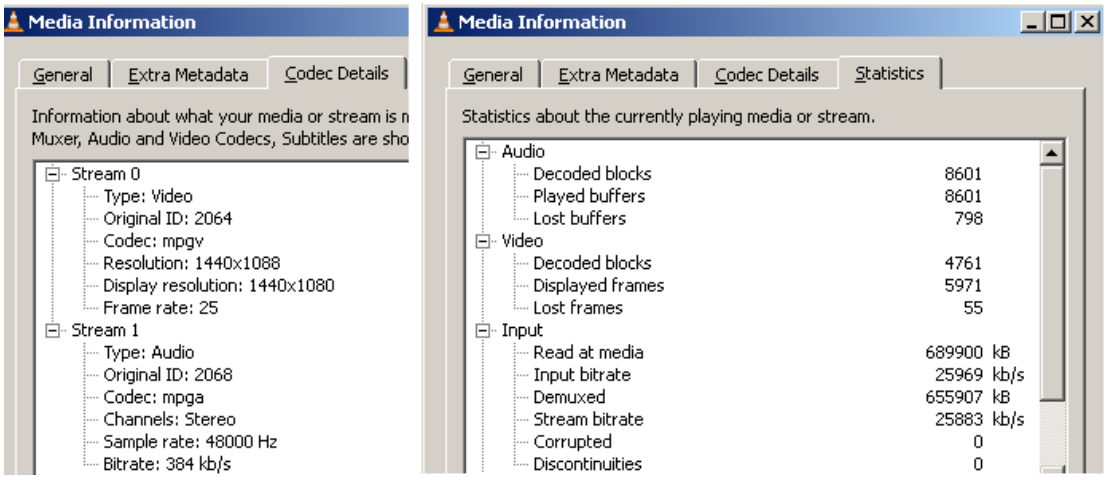


Figure 4.11: Details about the captured HD resolution and video bitrate

The raw movie takes a considerable amount of space comparing to its short duration. Before streaming the video, it needs to be compressed and hinted. We will compress the video so the size and the bitrate will be reasonably smaller and adequate for streaming while keeping the video and sound qualities as good as possible.

Also, note that any video file planned to be streamed using QuickTime Streaming Server or DSS must be hinted, i.e. they need a hint track for every streamable media track. The hint tracks tell the server exactly how to package the media data for the network; they give the server software pointers to the RTP information in order to serve the relevant media chunks. This information allows the server to deliver the correct video material in the sequence stipulated in the track file, and at the correct rate for the player display [4].

For compressing and hinting the video file in this experiment, Apple's QuickTime Player Pro which is cheap commercial software will be used. A step by step guide describing this process is explained in *Appendix E*.
Once the steps in *Appendix E* are completed, save your file. It will take some time before the file is compressed and hinted. For a 100 second long video file, and on a laptop equipped with a 32 bits Windows Vista operating system, a 1.9 GHz AMD Athlon X2 Dual-Core processor, 3 GB memory, this operation took around 30 minutes, which is 18 times more than real time compressing for the same file.

The output compressed file is now ready and details can be checked by the *Movie Inspector* feature in QuickTime Player. The video details are described in Figure 4.12.

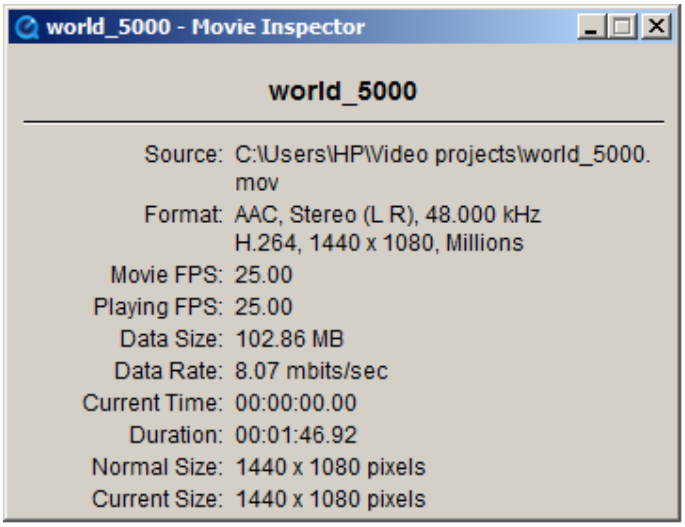


Figure 4.12: Compressed video files details

To stream the video file using DSS, it needs to be put in the media folder for streaming in DSS, here it was set to the default directory, i.e. in *C:\Program Files\Darwin Streaming Server\Movies*.

To view the movie as a stream, open QuickTime Player or VLC on a remote computer and enter the URL for the stream, meaning the RTSP address containing the IP address for the DSS server followed by the name of the file to be streamed.

For example, in VLC media player, go to Media, Open Network Stream and choose RTSP protocol, enter the address and then click Play. See Figure 4.13.

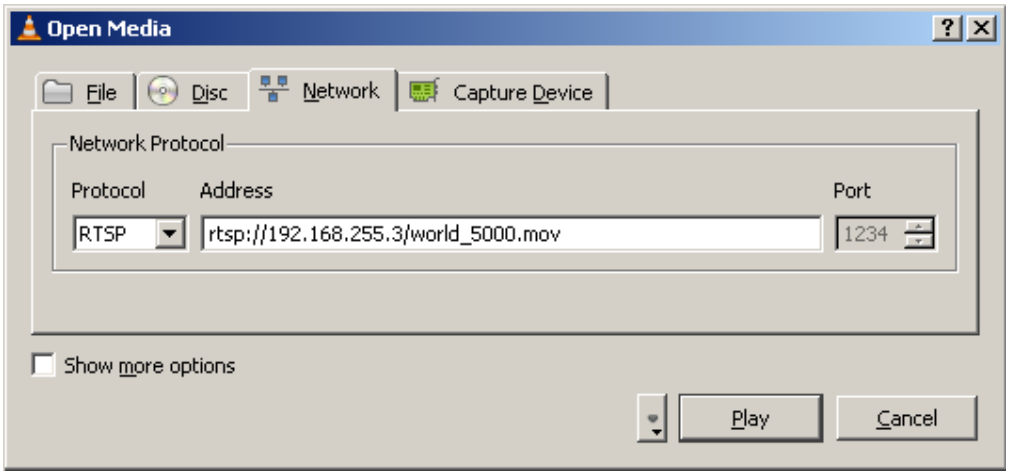


Figure 4.13: Opening the stream in VLC media player

It only takes a few seconds and the video stream starts to play nicely in VLC on the remote computer. While playing, you can check the video information by opening Tools menu and then Media Information. The details for the streamed video are shown in both Figure 4.14 and Figure 4.15.

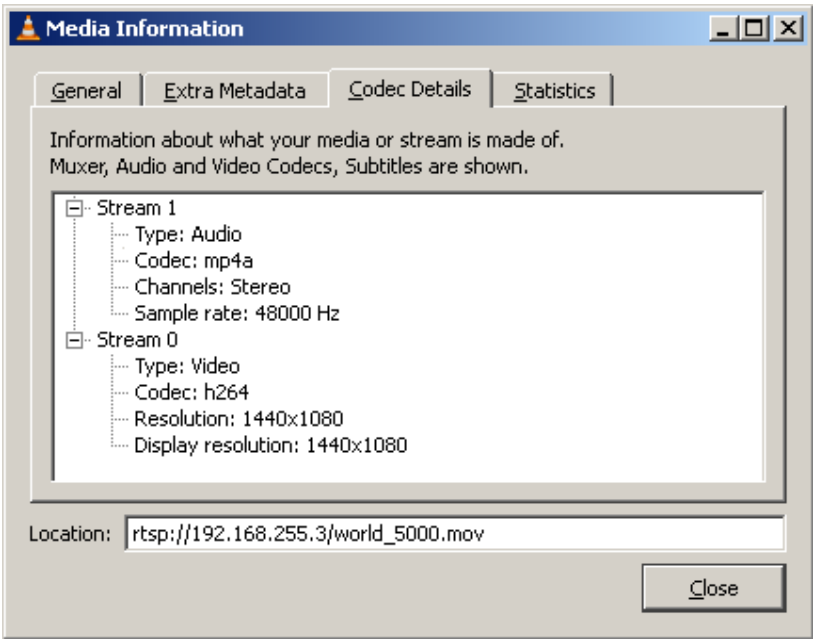


Figure 4.14: Codec details of the video

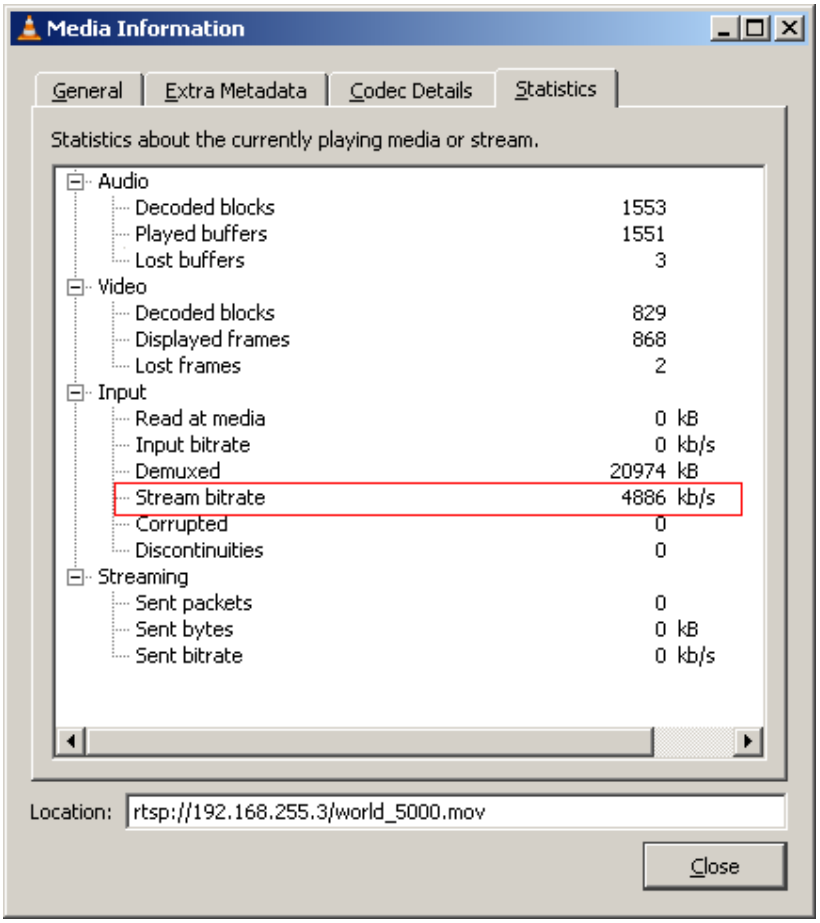


Figure 4.15: Bitrate details of the video

On DSS server side you can check the connected computer, which is currently playing the stream, from DSS administrator page by clicking the Connected Users link in the left menu.

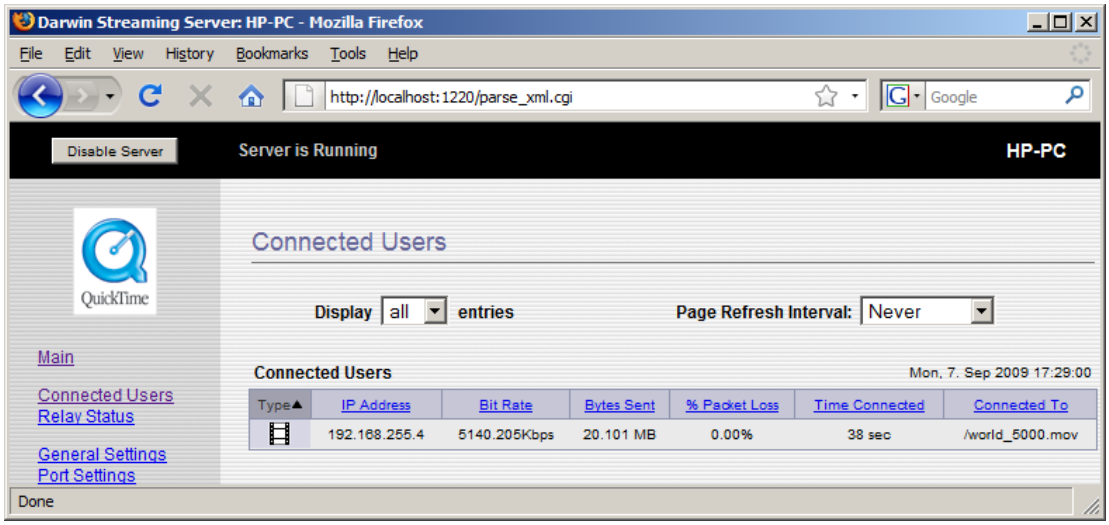


Figure 4.16: Connected users shown in DSS administrator page

4.3 Embedding the Movie into a Web Site

Another alternative to deliver the offline HD video is to include a QuickTime streaming movie into a Web site. To do so, a “reference movie” should be created. The reference movie is a small file that points to the actual movie to be streamed, and it helps to launch QuickTime Player Plug-in into the browser. See *Appendix F* on how to create a reference movie.

Once the movie is created, it should be saved in the same directory as the movie referenced to.

To embed the movie into a Web page, it is enough to transfer the reference movie and the HTML file that includes the necessary code to the Web server. Here is a sample code, which was used for embedding the video into the Web page.

```
<HTML>
<BODY>
<br>
<br>
<br>
<B><FONT COLOR="black">Click the QuickTime Player icon to play the movie</FONT></B>
<br>
<br>
<br>
<object classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B" width="72" height="80"
  codebase="http://www.apple.com/qtactivex/qtplugin.cab">
 <param name="src" value="image.jpg">
 <param name="href" value="HB_5000_ref.mov">
 <param name="type" value="video/quicktime">
```

```
<param name="target" value="quicktimeplayer">
<embed src="image.jpg" href="HB_5000_ref.mov target=quicktimeplayer
type="video/quicktime" width"72 height="80" autoplay="true" controller="true
pluginspage="http://www.apple.com/quicktime/download/">
</embed></object>
</BODY>
</HTML>
```

The *classid* attribute uniquely identifies the player software to use. It must be always set to "clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B". This unique code identifies an ActiveX control that must be installed on the user'Ats PC before the movie can be played. If the user does not have the ActiveX control installed, the browser can automatically download and install it.

The *codebase* parameter tells the browser where to find the ActiveX control for downloading if not already installed on the system. This parameter will point to a URL which will always have the latest version of the QuickTime ActiveX control.

The *embed* element is used for embedding the actual video to be played which will be automatically played if *autoplay* value is set to *true*.

The *pluginspace* attribute allows specifying a URL from which the user can fetch the necessary plug-in if it is not installed. This attribute is handled by the browser. If the browser can not find the plug-in when loading your page, it will warn the users and allow them to bring up the specified URL, from which one could download QuickTime which includes the QuickTime Player plug-in.

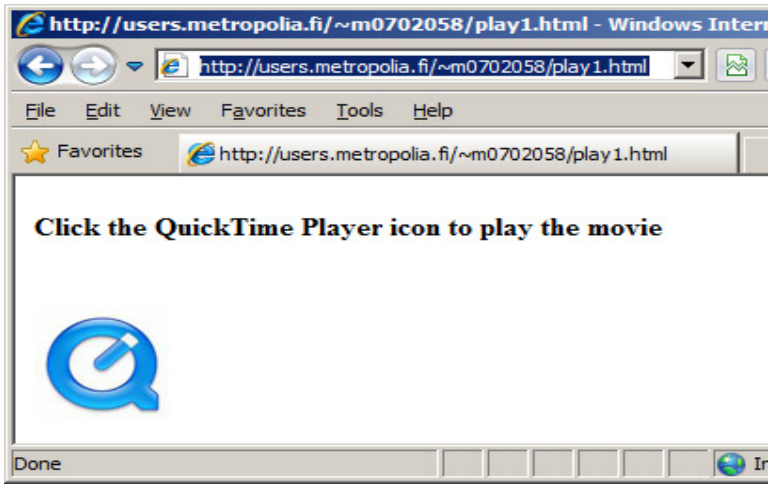


Figure 4.17: Embedding QuickTime Player in the browser to play the stream

By clicking the icon, the QuickTime Player will be launched to play the stream.

Figure 4.18: HD video streaming at 1280x720 resolution

5. PROPOSAL FOR LIVE STREAMING

In order to be able to make a live HD streaming across a LAN, hardware encoders must be involved to make the compression for real time HD content possible. Such hardware can make on the fly encoding while streaming the live video simultaneously. At the time of writing this thesis, only a few manufacturers were offering some solutions that, according to them, can serve for delivering real time streaming for high definition videos. Due to the high cost of those solutions it was impossible to do the experiment to test the live HD streaming in the school's laboratory. However, a proposal for making such experiment possible is introduced in this chapter.

After making a bit of market and Internet research, a few hardware-based solutions were found. The common factor between all those solutions is the cost: they are much too expensive for making an experiment for this study; they are rather used by professionals for commercial purposes. Below is a list of a couple of hardware devices which are relatively cheap but capable of achieving the same goal.

5.1 Hardware H.264 Coding Equipment

VMC-7440 HD-SDI is a compression board manufactured by VITEC Multimedia which delivers real time HD AVC/H.264 up to 1080p [16]. It costs about 3000 €. This is the cheapest hardware solution in the market for encoding HD content using H.264. However, the main drawback of this board is that it is Windows dependent and it is not compatible with other operating systems. Also, as an integrated board, it makes it vulnerable to conflicts between other Windows components and it can be easily outdated and become incompatible with the operating system in case of Windows updates or upgrades.

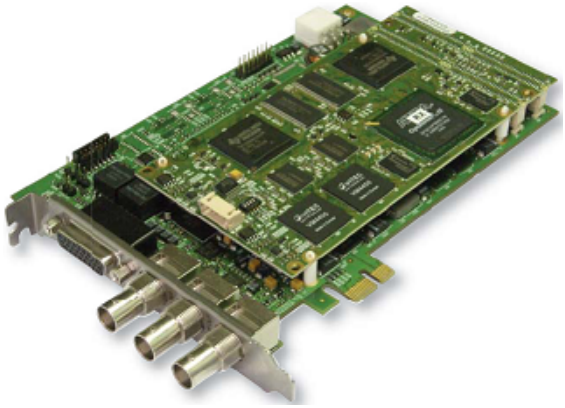


Figure 5.1: VMC-7440 HD-SDI compression board

A conceptual drawing for live HD video streaming using VMC-7440 HD-SDI compression board is shown in Figure 5.2.

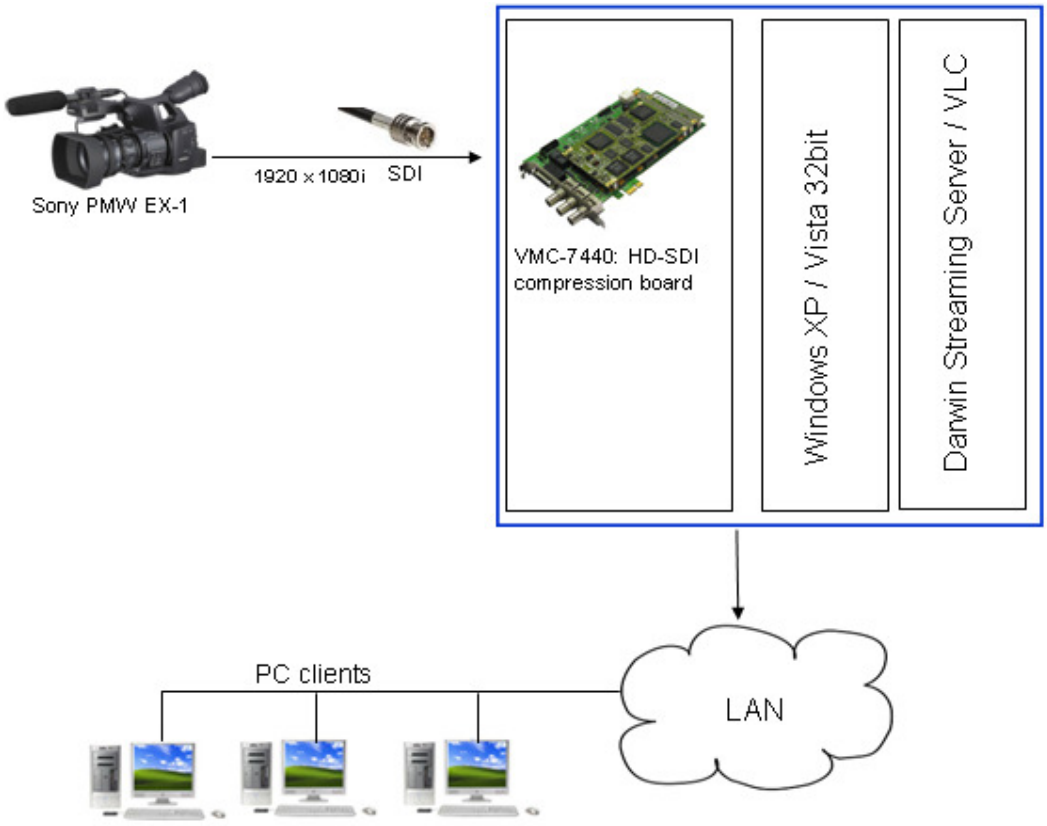


Figure 5.2: Using VMC-7440 HD-SDI for live HD streaming

5.2 Live Streaming System Design Option

Spinnaker 7000 is another solution for live HD encoding that supports not only H.264 but also VC-1 and Flash VP6 codecs. It is an external device manufactured by Inlet Technologies, which is specialized in making advanced encoding solutions for delivering multimedia content over IP networks.

Spinnaker 7000 is rather an expensive device, it costs about 13000 € from the reseller in Finland, but is still less expensive than a few other solutions in the market. The main advantages about Spinnaker 7000:

- Easy to set up: simple to connect to the video source and to the network
- Multi format: supporting VC-1, Flash VP6 and H.264 codecs
- Supported resolutions from mobile to web to HD

- Platform independent: can be used with different operating systems

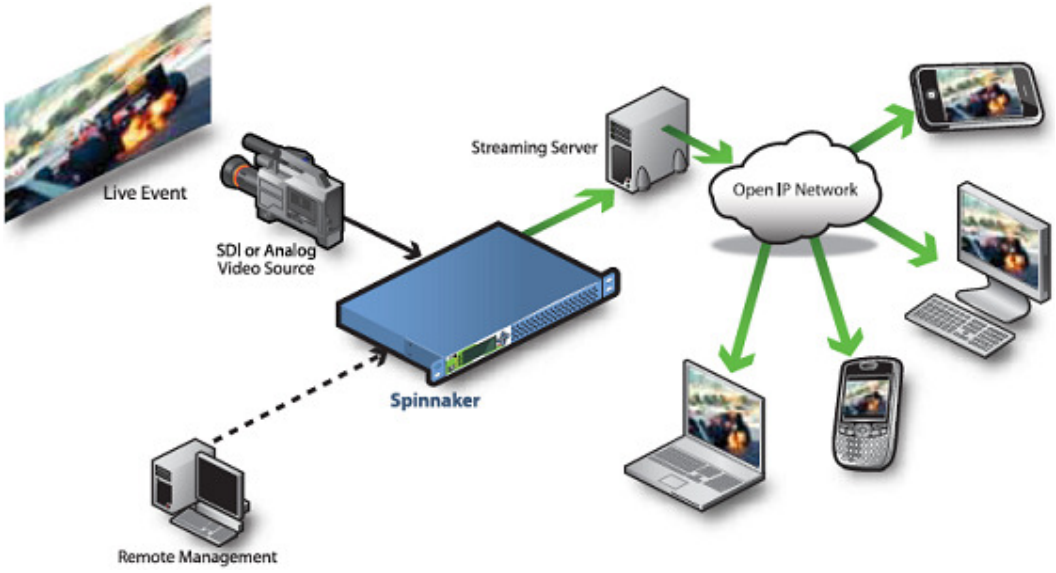


Figure 5.3: Set up of Spinnaker for live streaming over the network [17]

6. RESULTS AND CONCLUSION

For analyzing the quality of video streaming, two high definition videos at 1440x1080 and 1280x720 resolutions and encoded with 5 Mbps bitrate each were used. Both streaming inside a LAN and over the Internet were analyzed.

For streaming inside the LAN, Darwin Streaming Server installed on a home machine was used for testing, see chapter 4 for details. As for streaming over the Internet, a comparison between streaming from a home streaming server and streaming from a Content Delivery Network (CDN) was conducted. For the CDN, we used a commercial hosting server offering streaming capability using QuickTime Streaming Server installed on Mac OS operating system. Wireshark was used to analyze and compare the end user streaming quality.

6.1 Streaming inside a LAN

As discussed in chapter 4, Darwin Streaming Server installed on a home machine was used to stream to another client located inside the same LAN. The streaming

quality received by each client inside the LAN was excellent without any packet losses for two different videos at 1280x720 and 1440x1080 resolutions, encoded with 5 Mbps bitrate each. Figure 6.1 shows the stream received by the client while Figures 6.2 and 6.3 show, using Wireshark, the average amount of packets received per second as a graphical interpretation of the packets receives and function of time.

Figure 6.1: the stream as received by the client located within a LAN

Traffic	Captured	Displayed	Marked
Packets	21894	21894	0
Between first and last packet	41.680 sec		
Avg. packets/sec	525.283		
Avg. packet size	1193.890 bytes		
Bytes	26139038		
Avg. bytes/sec	627130.133		
Avg. MBit/sec	5.017		

Figure 6.2: Statistics summary of the stream using Wireshark

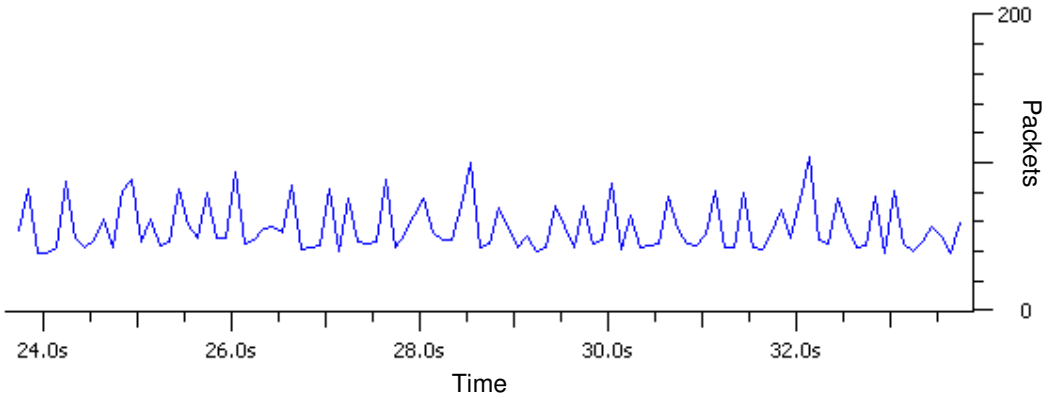


Figure 6.3: Number of received packets in function of time

6.2 Streaming over the Internet

6.2.1 Video at 1280x720 resolution with 5 Mbps bitrate

6.2.1.1 Streaming over Internet using DSS installed on a home server

The streaming server (DSS) was installed on a home server, using a laptop equipped with a 32 bits Windows Vista operating system, a 1.9 GHz AMD Athlon X2 Dual-Core processor, 3 GB memory. The Internet connection was 5 Mbps/1 Mbps maximum speed. When a client plays the stream over the Internet, the quality of received video suffers from a notable loss of packets as shown in Figure 6.4.

Figure 6.4: The stream at 1280x720 as received on the client side

In DSS and from *Connected Users* menu, you can see clearly the amount of packets lost in terms of percentage during the streaming. See Figure 6.5.

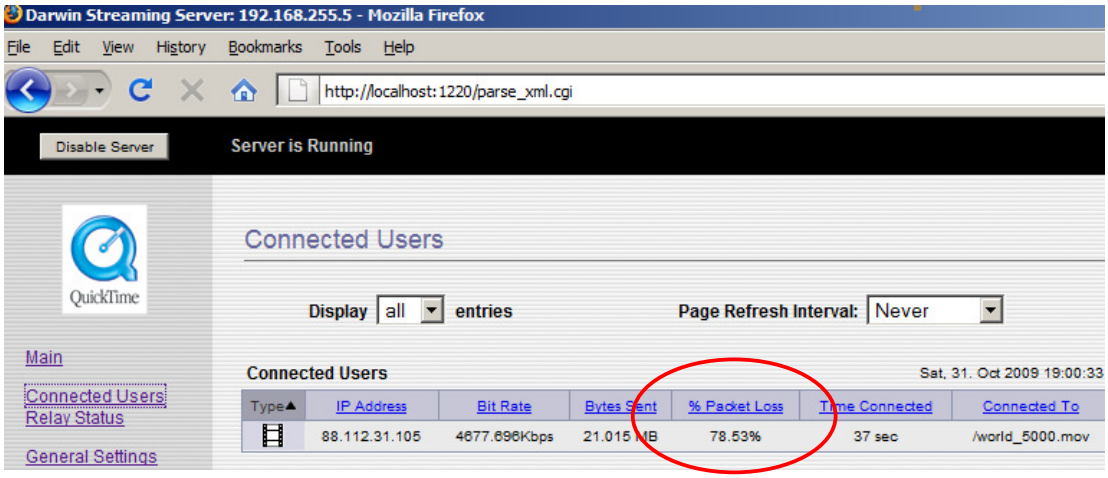


Figure 6.5: Packet loss information as shown on DSS

We used Wireshark *Statistics* feature to examine the amount of packets received and the average bitrate at the client side. As shown in Figure 6.6, the average bitrate was dramatically dropped leading to a very poor quality of the stream.

Traffic	Captured	Displayed	Marked
Packets	4376	4376	0
Between first and last packet	38.232 sec		
Avg. packets/sec	114.460		
Avg. packet size	1107.899 bytes		
Bytes	4848165		
Avg. bytes/sec	126810.596		
Avg. MBit/sec	1.014		

Figure 6.6: Statistics summary of the stream using Wireshark

Also, we made use of the *IO Graphs* feature from *Statistics* in Wireshark to represent the findings in graph mode as shown in Figure 6.7. The graph shows the amount of packets received by the client in function of time. It is clearly shown that the number of the packets is frequently dropped and even lost.

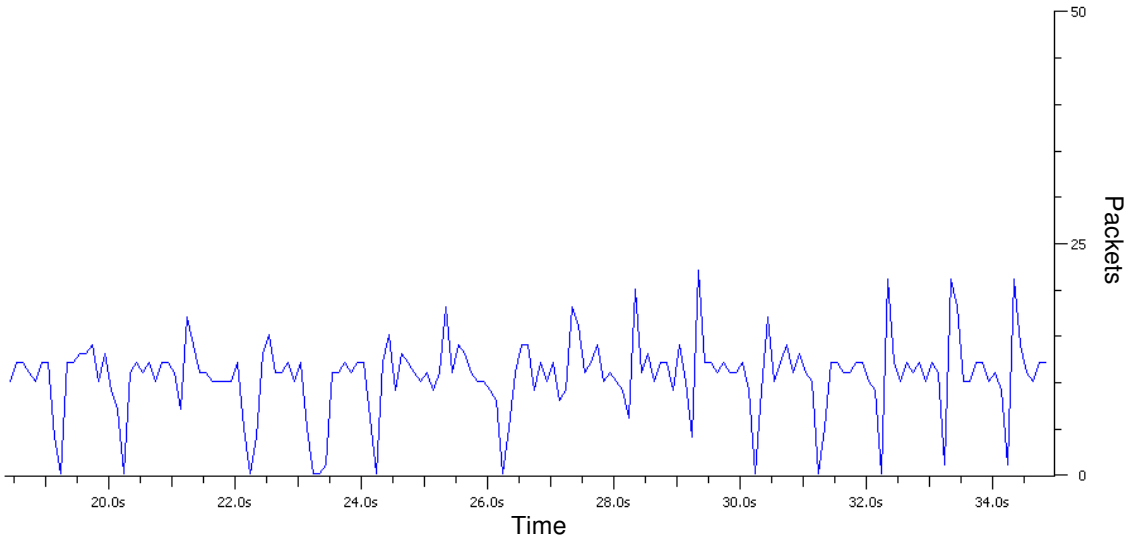


Figure 6.7: Number of received packets in function of time

6.2.1.2 Streaming over Internet using QuickTime Streaming server on a CDN

In order to test high definition streaming from a more powerful server, with high Internet speed and high bandwidth connections to the streaming clients, a commercial streaming video web hosting server with QuickTime Streaming Server (QSS) is used. The hosting service offers many bandwidth options, ranging from 5 Gb/s to 100 Gb/s to choose from, depending on the size of the audience receiving the streams. As the service is planned for thesis and testing purposes, a 5 Gb/s bandwidth was chosen as the audience contains only a couple of persons to check the streaming.

When playing the RTSP stream directly from Quick Time player, either as stand alone or as embedded into a web site, it plays smoothly with an outstanding video and audio quality. Figure 6.9 shows a screen shot from a video as received at the client's side.

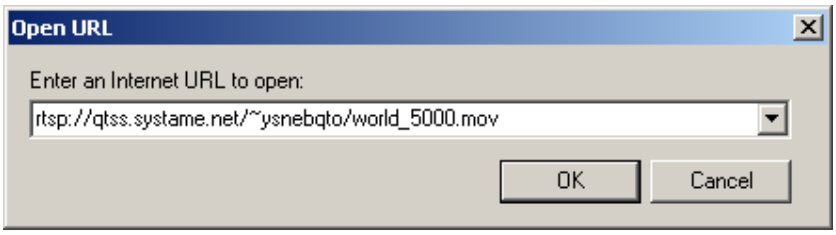


Figure 6.8: Playing the stream from a web hosting server using QuickTime Player

Figure 6.9: The stream at 1280x720 as received on the client side

As was done for streaming from a home server, we used the Wireshark *Statistics* feature to examine the amount of packets received and the average bitrate at the client side. As shown in Figure 6.10, the original bitrate is kept and the average of packets received per second increased about 8 times which is a significant improvement comparing to playing the movie from a home server as described earlier.

Traffic	Captured	Displayed	Marked
Packets	34202	34202	0
Between first and last packet	39.724 sec		
Avg. packets/sec	860.984		
Avg. packet size	803.793 bytes		
Bytes	27491320		
Avg. bytes/sec	692052.437		
Avg. MBit/sec	5.536		

Figure 6.10: Statistics summary of the stream using Wireshark

Again, we used the *IO Graphs* feature from *Statistics* in Wireshark to represent the findings in graph mode as shown in Figure 6.11. The graph shows the amount of packets received by the client in function of time. The amount of packets per seconds received by the client is quite steady and remains within a tolerable range.

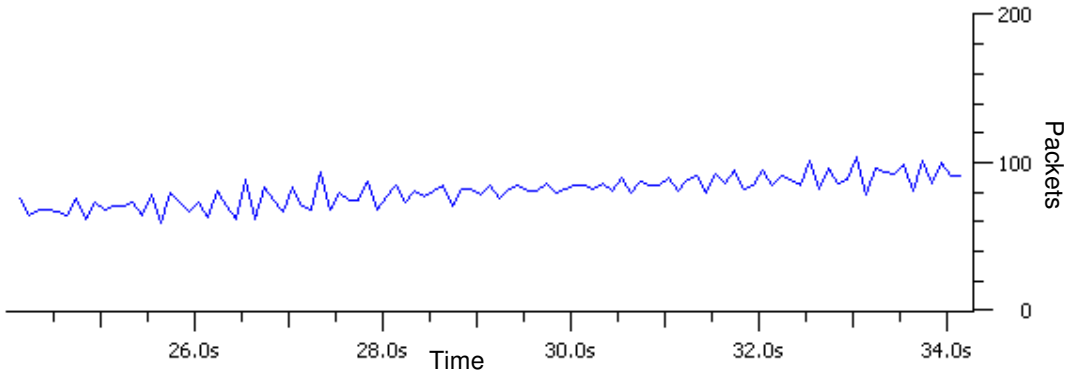


Figure 6.11: Number of received packets in function of time

6.2.2 Video at 1440x1080 Resolution with 5 Mbps Bitrate

For analyzing a video with a higher resolution, at 1440x1080, the same approach done for 1280x720 video is followed.

6.2.2.1 Streaming over Internet Using DSS Installed on a Home Server

A screen shot from the video received at the client's side is shown in Figure 6.12.

Figure 6.12: The stream at 1440x1080 as received on the client side

Figures 6.13 and 6.14 show the average packets per second received as well as for bitrate. Again, the packets and bitrate losses are clearly visible.

Traffic	Captured	Displayed	Marked
Packets	9602	9602	0
Between first and last packet	69.659 sec		
Avg. packets/sec	137.843		
Avg. packet size	945.704 bytes		
Bytes	9080650		
Avg. bytes/sec	130358.402		
Avg. MBit/sec	1.043		

Figure 6.13: Statistics summary of the stream using Wireshark

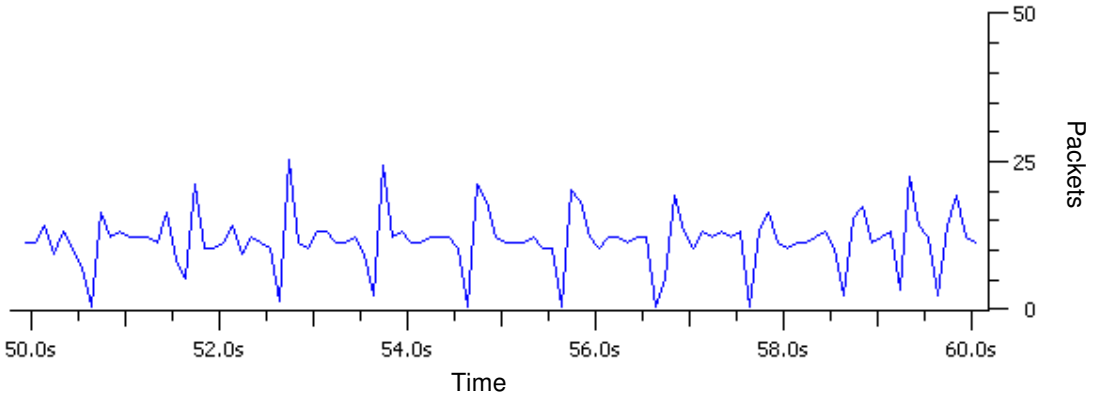


Figure 6.14: Number of received packets in function of time

6.2.2.2 Streaming over Internet Using QuickTime Streaming Server on a CDN

The quality of the stream received by client when streamed from a video streaming web hosting was excellent, with hardly any packet or bitrate losses.

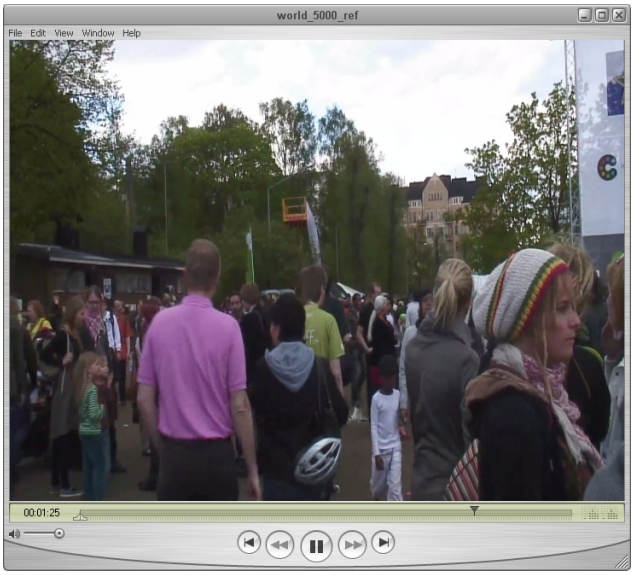


Figure 6.15: The stream at 1440x1080 as received on the client side

Traffic	Captured	Displayed	Marked
Packets	31271	31271	0
Between first and last packet	68.558 sec		
Avg. packets/sec	456.126		
Avg. packet size	1180.121 bytes		
Bytes	36903565		
Avg. bytes/sec	538284.099		
Avg. MBit/sec	4.306		

Figure 6.16: Statistics summary of the stream using Wireshark

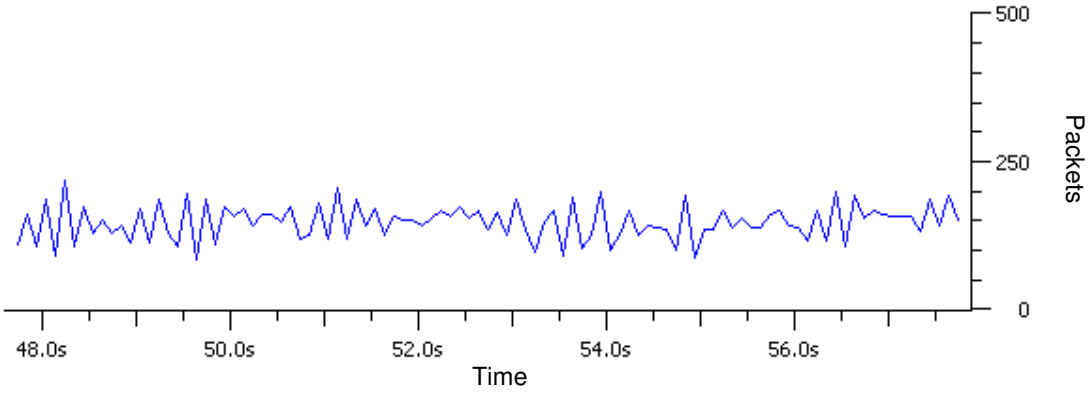


Figure 6.17: Number of received packets in function of time

7. SUMMARY

The objective of this thesis was to describe high definition video streaming using H.264 compression. The initial plan was also to carry out live HD streaming experiment but this turned out to be impossible as the cost for the hardware needed was too expensive for thesis work such as this. Instead a proposal for making live HD video streaming possible was introduced.

An introduction about streaming, streaming architecture and technologies was covered. Also a brief overview on H.264 video codec and high definition technology was presented.

The detailed experiment steps about offline video streaming were described. The streaming was done first inside a LAN by setting the streaming server on a home machine and encoding the video using H.264 codec. QuickTime Player Pro tool was used to perform the video and audio encoding step. The streaming was successful except for some minor issues with firewalls which were overcome by adding the ports to the firewall exceptions on the streaming server. However, there were some big issues when streaming beyond the LAN over the Internet. When using a home server to stream over the Internet, the main issues were created by the firewalls as many of them are configured to restrict the streaming packets by port numbers. Sometimes, it was not enough to only add the ports to the firewall exceptions but NAT settings and port forwarding in router configuration was needed (when a router is used). Another major issue, which has largely affected the quality of streaming, is the speed of Internet and the bandwidth size. For this thesis, both Internet speed and bandwidth provided by the Internet Service Provider (ISP) were not enough to provide proper video streaming to the clients. A commercial web hosting service, offering video streaming on a QuickTime Streaming Server (QSS), was used. By offering a 5 Gb/s large bandwidth, streaming works without any problem provided that the clients have enough Internet speed to receive the encoded stream.

For the analysis of the streams, Wireshark was used to look at different outputs in terms of packets received per seconds. This was done by streaming two high definition videos at different resolutions and using DSS streaming server installed

on a home machine and QSS hosted on a streaming video web hosting server. The results were presented and compared.

There are streaming quality metrics that need to be considered for providing adequate Quality of Service (QoS) for HD video streaming. QoS relates to the ability to get packets through the Internet reliably, with time-sensitive packets (such as those comprising media streams) able to jump queues and preallocate or reserve network resources, to avoid congestion. It also refers to the ability to guarantee time-sensitive packet delivery, with no packet loss [5].

To improve the QoS of high definition streaming both from the server and the client's sides, one needs to encode the video in H.264 using the rights settings depending on the tool used and a reasonable bitrate that make the difference for the quality of the video. Also, the right firewalls ports need to be open for the streams and a large bandwidth needs to be reserved from the server side as a fast broadband speed is needed on the client's side to handle the encoded stream without any noticeable packet loss. In addition, several tools exist for measuring the QoS for various metrics of video streaming, metrics such as connection time, buffer time, rebuffer events, protocols and route tracing.

REFERENCES

[1] Wes Simpson. 2008. Focal Press. *IPTV, Internet Video, H.264, P2P, Web TV, and Streaming: A Complete Guide to Understanding the Technology.*

[2] Keith Jack. 2005. Newnes. *Video Demystified: A Handbook for the Digital Engineer, Fourth Edition.*

[3] *AVC HD Paper*. Panasonic web pages. [WWW document] http://www.panasonic.com (Accessed 25.11.2009)

[4] David Austerberry. 2005. Focal Press. *The Technology of Video and Audio Streaming.*

[5] Michael Topic. 2002. Mc-Graw Hill. *Streaming Media Demystified.*

[6] Colin Perkins. 2003. Addison Wesley. *RTP: Audio and Video for the Internet.*

[7] Adobe Flash Media Server client-server architecture. Adobe Systems Incorporated. [WWW document] http://www.adobe.com/livedocs/flashmediaserver (Accessed 25.11.2009)

[8] RTMP specification. Adobe Systems Incorporated. [WWW document] http://www.adobe.com/devnet/rtmp (Accessed 27.09.2009)

[9] Joann Karas, Bachelor's Thesis, Live Streaming of Video to DVB-H Mobile Handsets. Helsinki Metropolia University of Applied Sciences, Stadia.2008.

[10] Flash Media Server Installation Guide. Adobe Systems Incorporated. [WWW document] http://livedocs.adobe.com/flashmediaserver/3.0/docs (Accessed 25.11.2009)

[11] VideoLAN web pages. [WWW document] http://www.videolan.org (Accessed 15.04.2009)

[12] The H.264/AVC Advanced Video Coding Standard. [WWW document] http://www.fastvdo.com/spie04/spie04-h264OverviewPaper.pdf (Accessed 25.11.2009)

[13] *H.264 Whitepaper*. AMD web pages. [WWW document] http://www.amd.com (Accessed 12.10.2009)

[14] *QuickTime and MPEG4: Now Featuring H.264*. Apple QuickTime web pages. [WWW document] http://www.apple.com/quicktime (Accessed 12.10.2009)

[15] Darwin Streaming Server Administrator. Apple Computer, Inc. [WWW document] http://developer.apple.com/opensource/server/streaming (Accessed 17.04.2009)

[16] VITEC Multimedia web pages. [WWW document] http://www.vitecmm.com/index.php (Accessed 08.10.2009)

[17] Inlet Technologies web pages. [WWW document] http://www.inlethd.com (Accessed 08.10.2009)

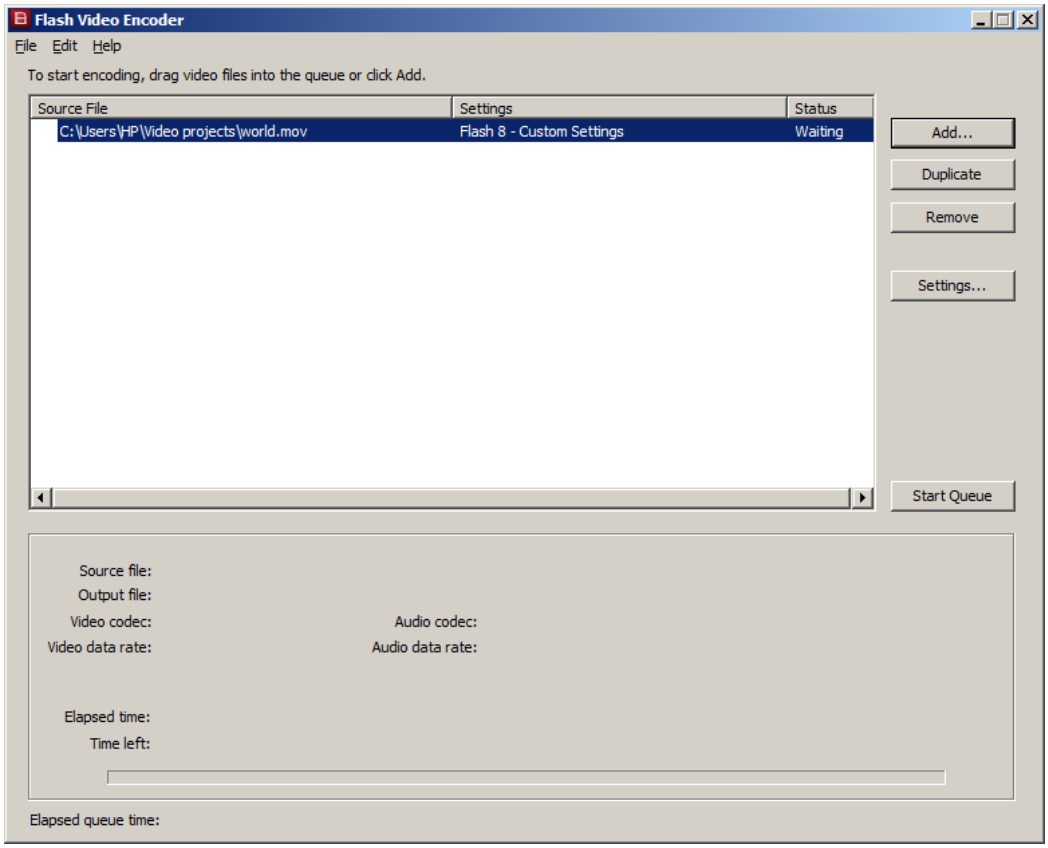
APPENDICES

APPENDIX A: RTP header main fields

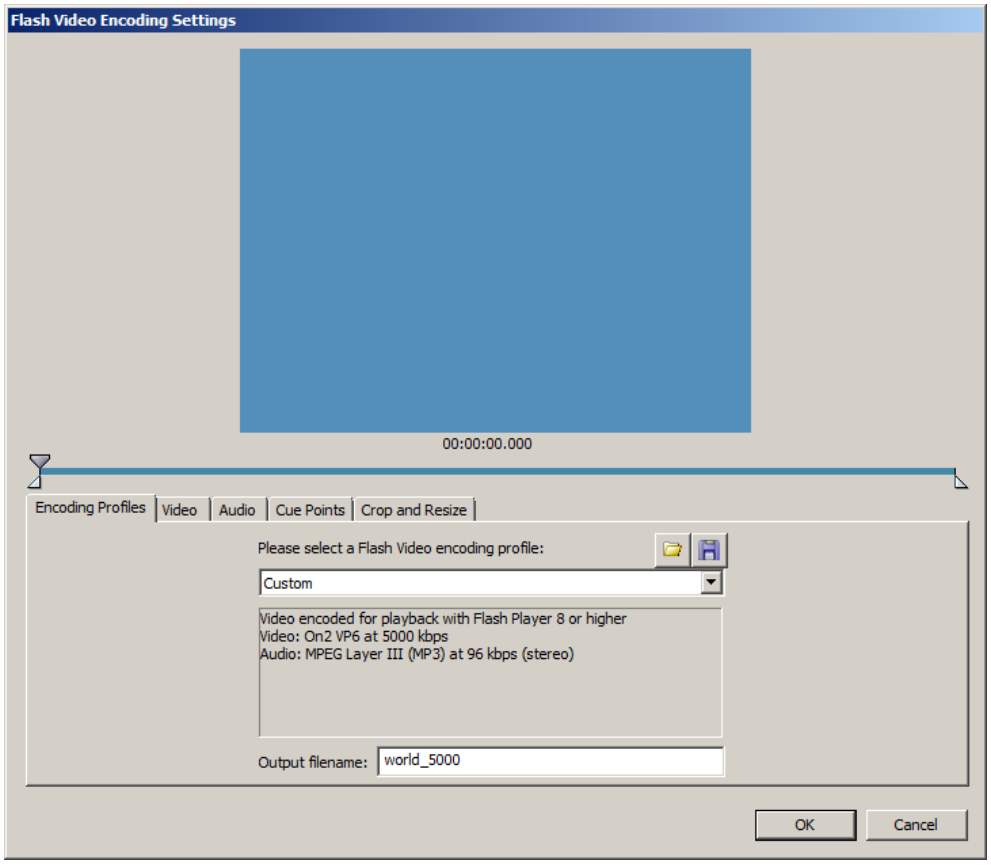
- **V**: Indicates the version of the RTP used.
- **P**: The padding bit, if set, indicates that there is one or more octets of padding present at the end of the packet which are not a part of the payload data.
- **X**: Indicates the extension. If the extension bit is set, then the RTP header is followed by one RTP Header extension. In practice, RTP header extensions are rarely used.
- **M**: Stands for the Marker. The marker bit usage is specified by the profile or application being used. Generally for video transmission applications, the M-bit signifies the end of data of one frame.
- **Payload Type**: 7 bits, providing 128 possible different types of encoding; e.g. PCM, MPEG2 video, etc. different media are not multiplexed.
- **Sequence Number**: 16 bits; random number incremented by one for each RTP data packet sent; used to detect packet loss.
- **Timestamp**: 32 bytes; gives the sampling instant of the first audio/video byte in the packet; used to remove jitter introduced by the network:
    - clock frequency depends on applications
    - random initial value
    - several packets may have equal timestamps (e.g. same video frame), or even in disorder (e.g. interpolated frames in MPEG)
- **Synchronization Source identifier (SSRC):** 32 bits; an id for the source of a stream; assigned randomly by the source.
- **CSRC**: Identifiers of the sources contributing to (mixed into) packet.

APPENDIX B: Encoding HD video using Adobe Flash CS3 Video Encoder

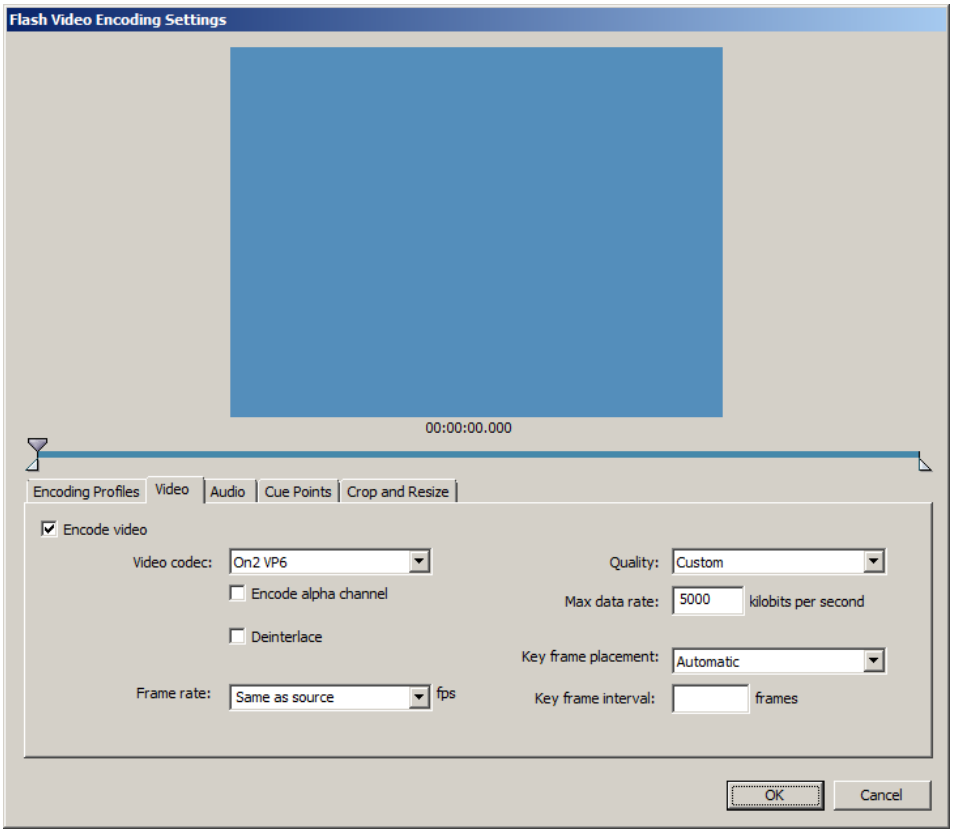
1. Open Adobe Flash CS3 Video Encoder and press *Add* button to *add* the video to encode.



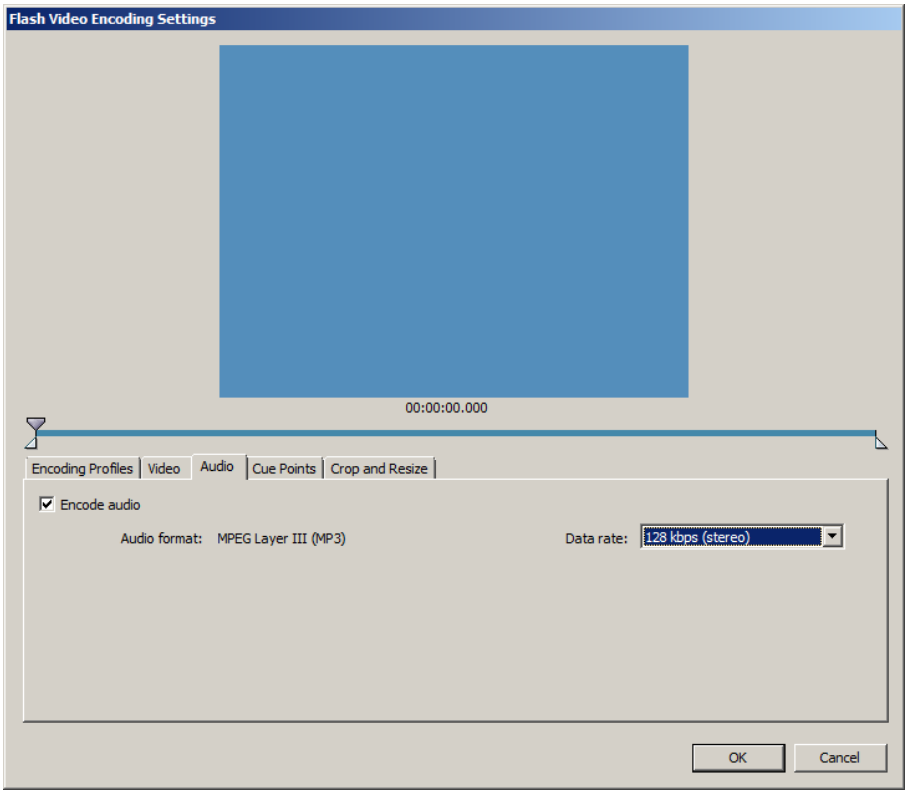
2. Activate the imported video and click *Settings* and type a name for the output file in *Output filename*.



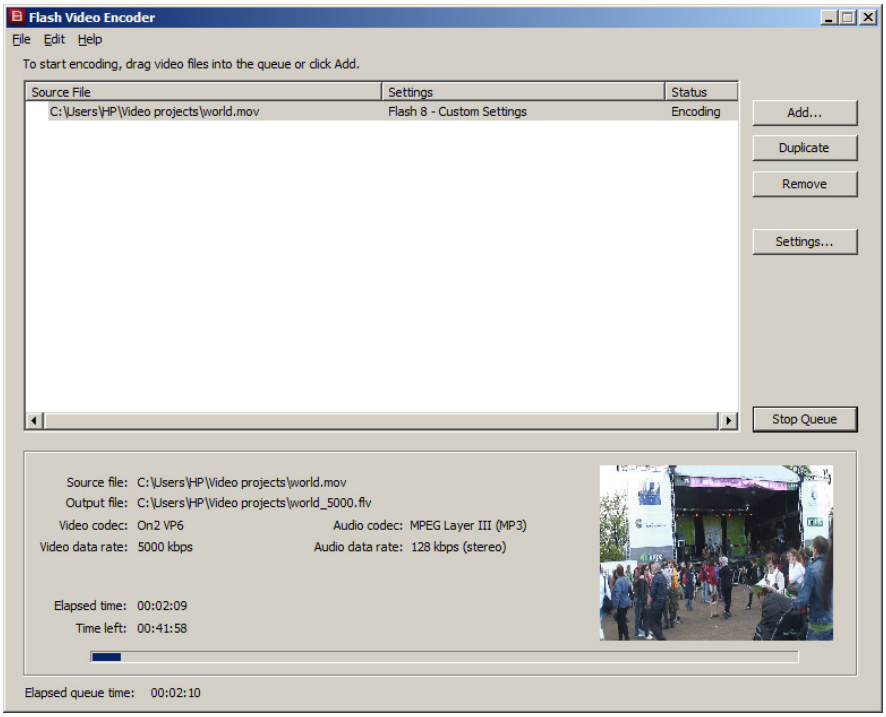
3. Click the *Video* tab, keep the default *One2 Vp6* codec as it generates much better image quality than *Sorenson Spark.* If your HD video was shot using 1080i then check *Deinterlace*, otherwise keep the default value. Enter the desired bitrate for encoding in *Max data rate* field.



4. Click *Audio* tab and choose 128 kbps (stereo) in *Data rate* and then click *OK*.



5. Click *Start Queue* to begin the encoding process.



APPENDIX C: SDP file parameters description

- **Session description**

  v= protocol version
  o= owner/creator and session identifier
  s= session name
  i= session information
  u= URI of description
  e= email address
  p= phone number
  c= connection information -not required if included in all media
  b= bandwidth information
  One or more time descriptions see next slide
  z= time zone adjustments
  k= encryption key
  a= zero or more session attribute lines
  Zero or more media descriptions see next slide

- **Time description**

  t= time the session is active
  r= zero or more repeat times

- **Media description**

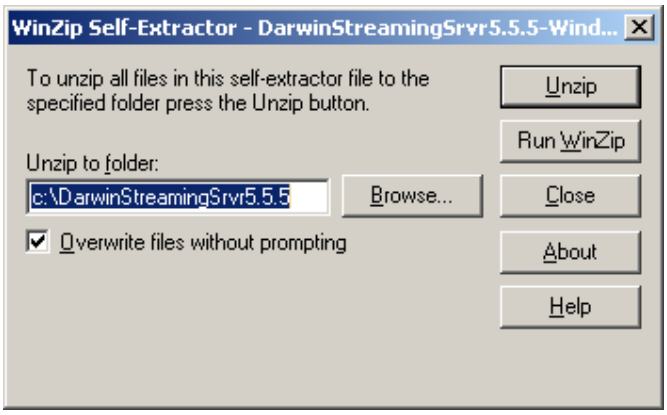
  m= media name and transport address
  i= media title
  c= connection information -optional if included at session-level
  b= bandwidth information
  k= encryption key
  a= zero or more media attribute lines

APPENDIX D: Darwin Streaming Server installation and configuration on Windows XP

1. Before installing Darwin Streaming Server, make sure to download and install the latest version of Active Perl from *http://www.activestate.com/activeperl/downloads*

2. Download Darwin Streaming Server from *http://dss.macosforge.org/post/previous-releases*. Scroll and click Streaming Server for Windows 2000/2003 Server under Windows installer (v5.5.5); the latest version available when writing this thesis.

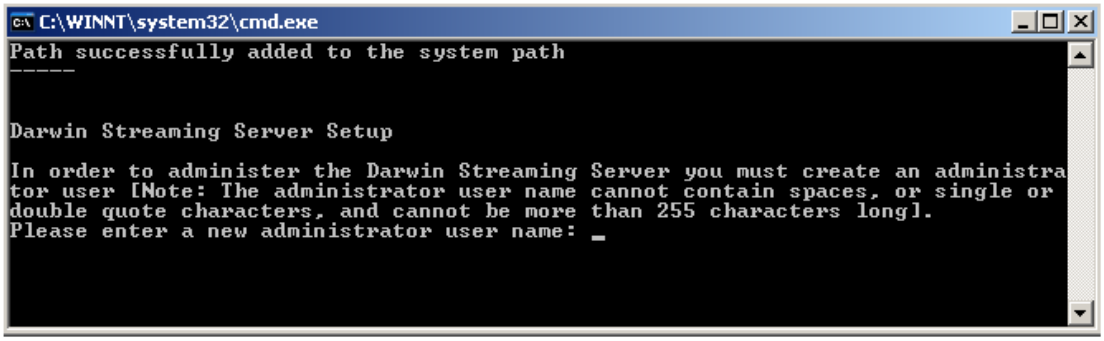
3. Double click the downloaded file to extract it to your local hard drive, e.g. to the default suggested folder (*C:\DarwinStreamingSrvr5.5.5*). The self-extractor window will open.



4. Close the self-extractor when the files are extracted.

5. Go to the folder where the files were extracted and run *Install.bat*. The DOS prompt is open and the installation starts. Make sure that there is no application running.

6. Once the installation is finished enter then the administrator user name and password when prompt.



The server should be now installed in *C:\Program Files\Darwin Streaming Server*

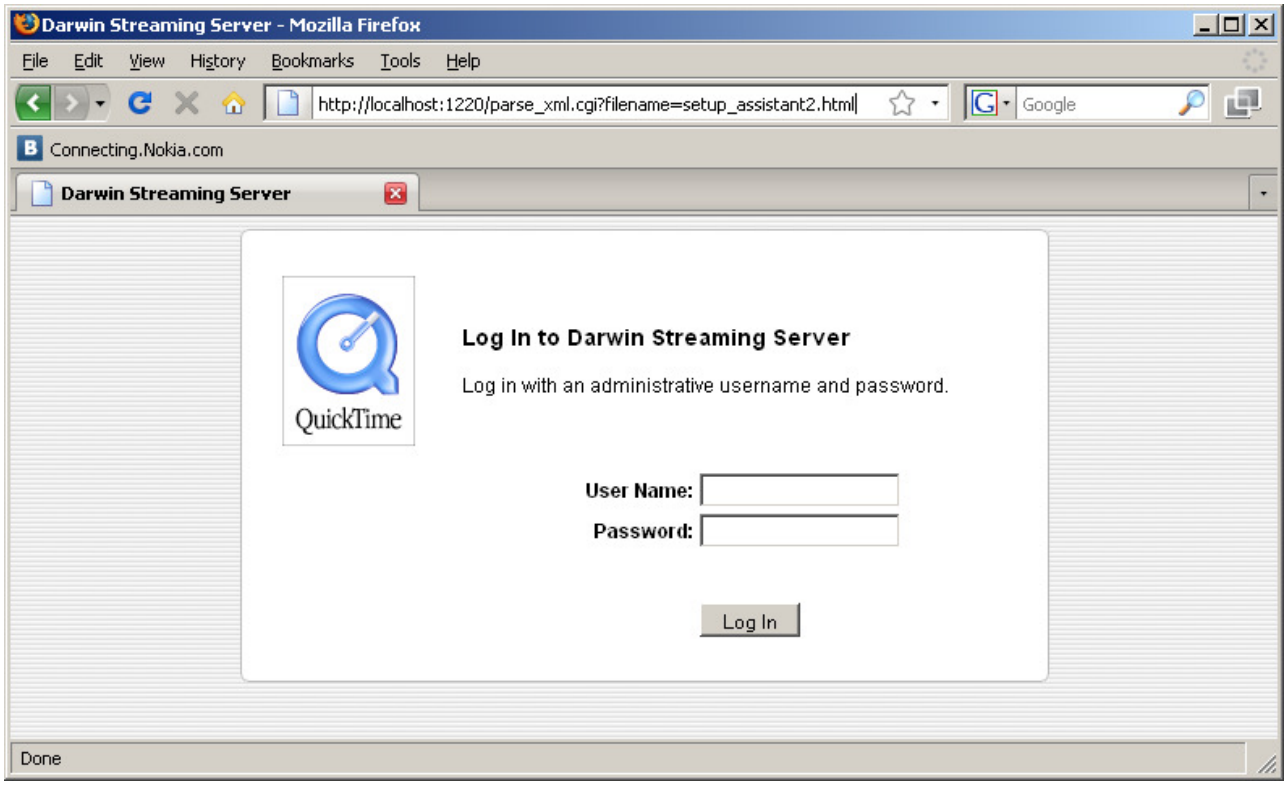
You can maintain your administrator user name and password by running the following command in DOS command prompt:

C:\Perl\bin\perl "C:\Program Files\Darwin Streaming Server\ WinPasswdAssistant.pl"

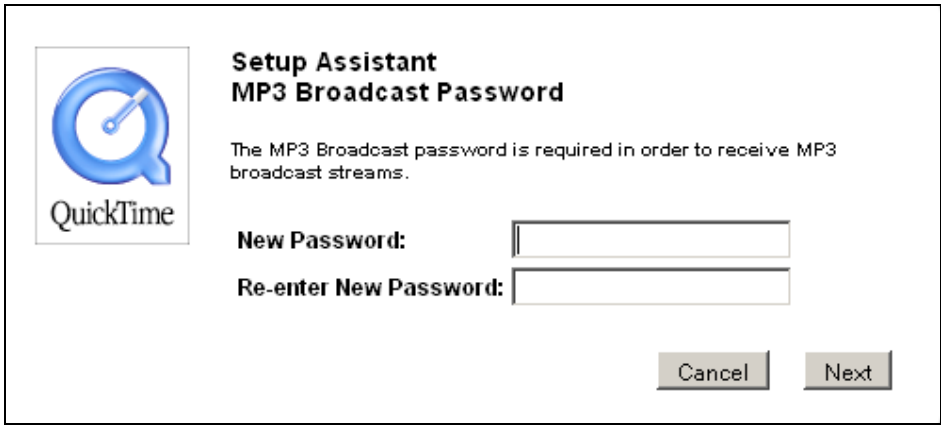
Note: this assumes that Active Perl is installed in *C:\Perl*

7. Test the installation by opening your browser and typing this address: *http://localhost:1220*

   Then enter your administrator user name and password. Click *Login*.



The setup assistant of Darwin Streaming Server will ask you to create an MP3 Broadcast Password:

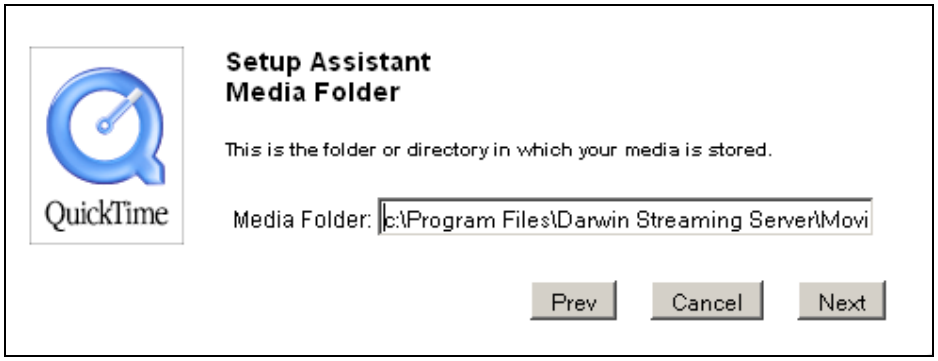


Enter your desired password and click *Next*.

Then you will be asked if an SSL certificate will be used:

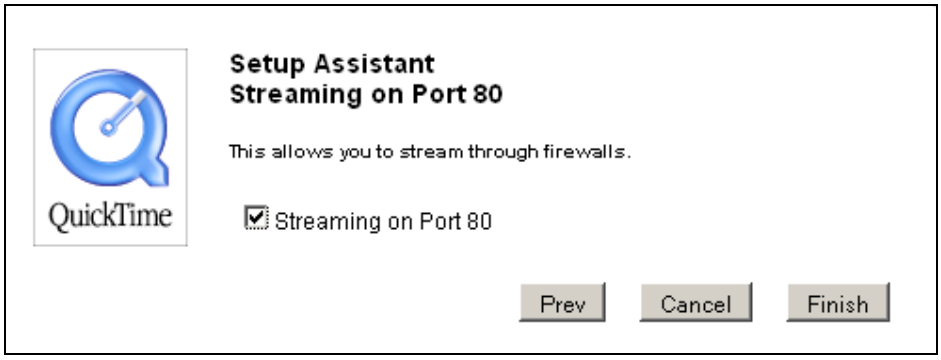
Using SSL certificate is outside the scope of this thesis. Do not tick the check box and click *Next*.

The next step is to set the media Folder for streaming. You can keep the default folder, i.e. C:\Program Files\Darwin Streaming Server\Movies



Click *Next*.

Finally set the streaming port to 80:



Click *Finish*.

Caution: If the port 80 is used by other web servers, such Apache and IIS, which are installed on the same computer as Darwin Streaming Server then DSS will not work.

APPENDIX E: Hinting and compressing the video file using QuickTime Player Pro

1. Open the video in QuickTime Player Pro, go to *File – Export* and type a new name for the file. For Export, choose Movie to QuickTime Movie and keep the *Default Settings* for *Use*.

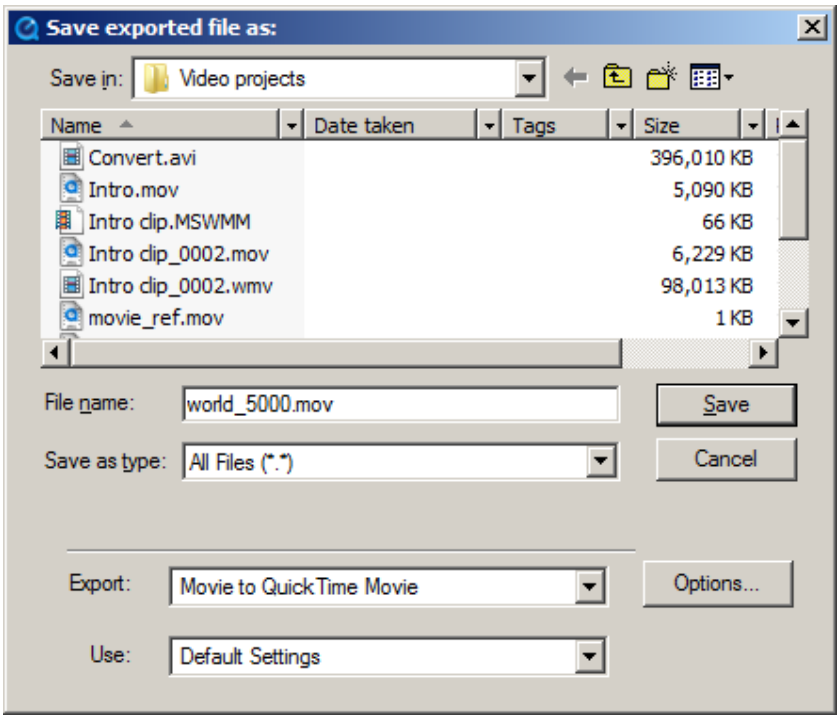


Figure E.1: Exporting a movie in QuickTime Pro

2. Click *Options* and then click *Settings* in Video. Choose H.264 as *Compression Type* as this codec will produce the best image quality while keeping the bitrate and the size of the video file small. Keep *Current* for *Frame Rate* and choose *Automatic* for *Key Frames*. For *Data Rate*, choose *Restrict to* and type 5000, which will be equivalent to 5 Mbps, an acceptable bitrate by most broadband connections. Then choose *Streaming* in the *Optimized for* drop down list. Under *Compressor*, choose *Best quality (Multi-pass)* for *Encoding* option. Click *OK* to close the video compression settings window. The video compression settings are illustrated in Figure E.2.

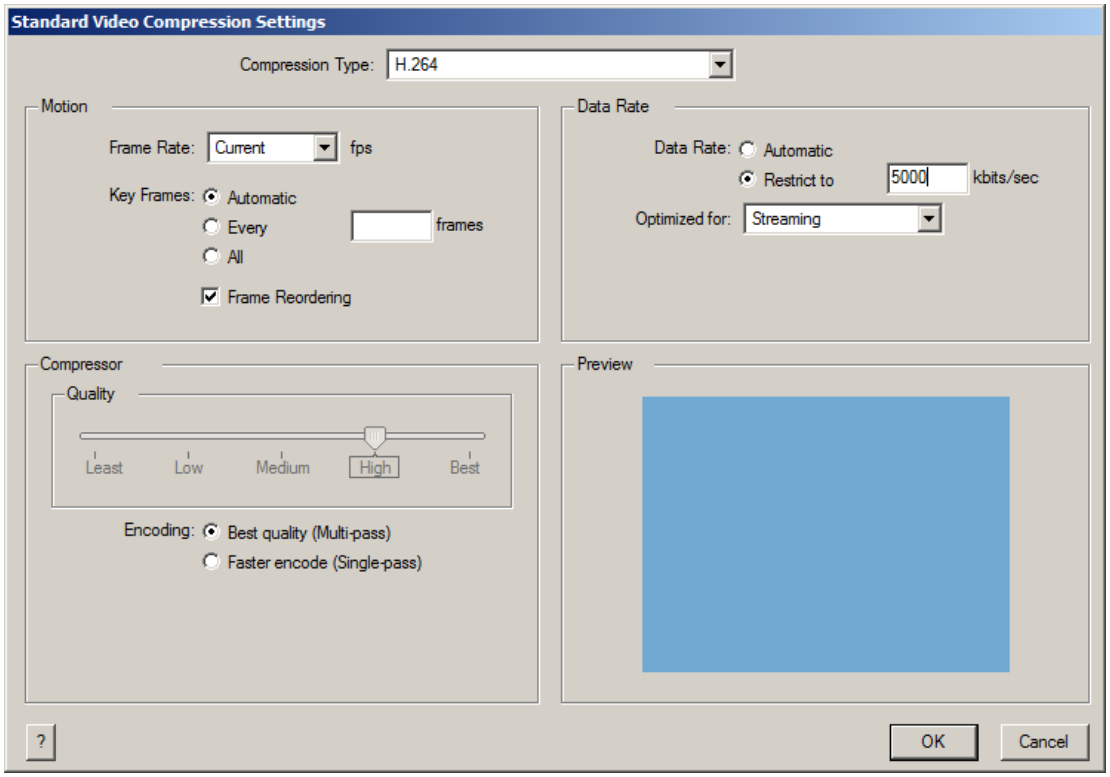


Figure E.2: Video compression settings for the video file

If the video is 1080i, click *Size* button and check *Deinterlace Source Video* tick box. If your video is in progressive format, e.g. 720p, you can skip this step. Click *OK*.

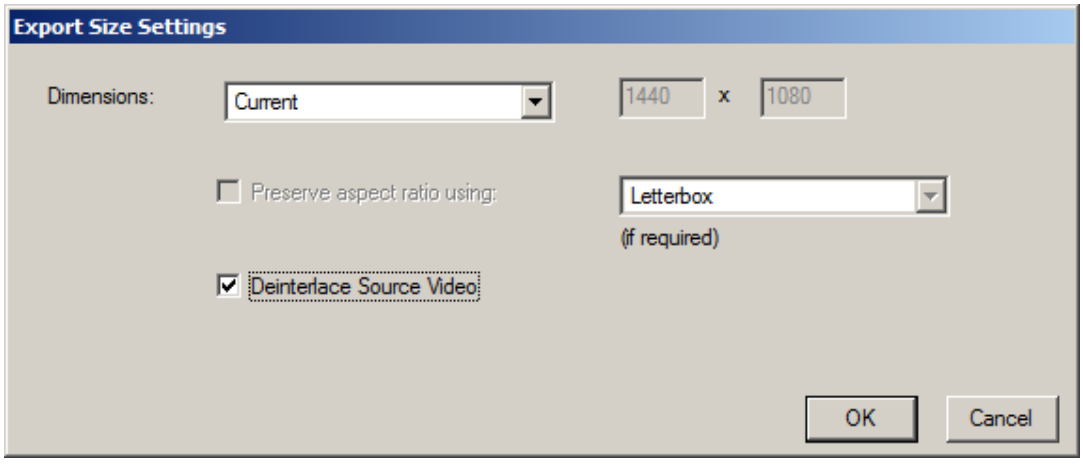


Figure E.3: Size Settings for the video file

Note: If the video is intended to be played back on a computer, you can deinterlace the video using the editing program or capture device. However, if the final video will be played on a standard television by a DVD, you need to maintain interlacing while you edit.

3. Under *Sound*, click *Settings* button. Choose *AAC* and then *Recommended* from the *Format* and *Rate* drop down lists respectively. For *Quality* choose *Best* and *Target Bit Rate* 128. Click *OK*. The sound settings are illustrated in Figure E.4.

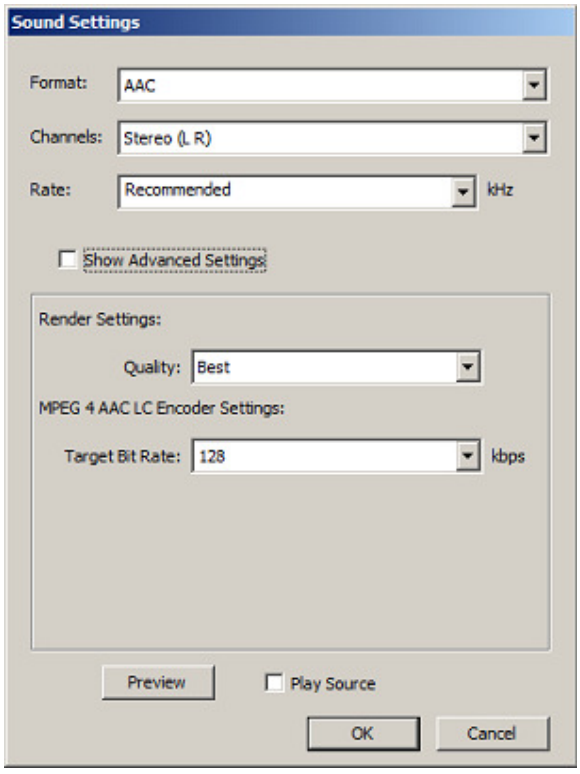


Figure E.4: Audio compression settings for the video file

Still in the *Movie Settings*, make sure that *Prepare for Internet Streaming* is checked and choose *Hinted Streaming* from the drop down list.

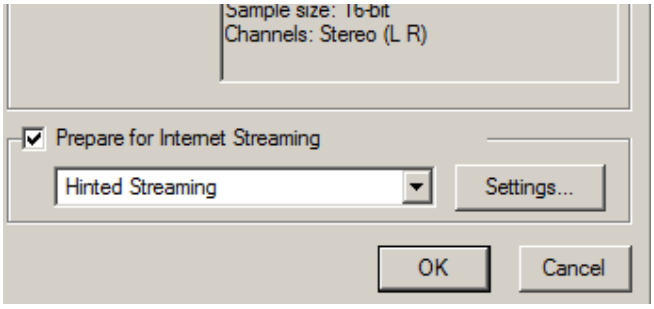


Figure E.5: Setting Hinted Streaming under Movie Settings

4. Click Settings button and check Optimize Hints for Server (create larger file). This will create a large file but the streaming will be much faster. Click OK.

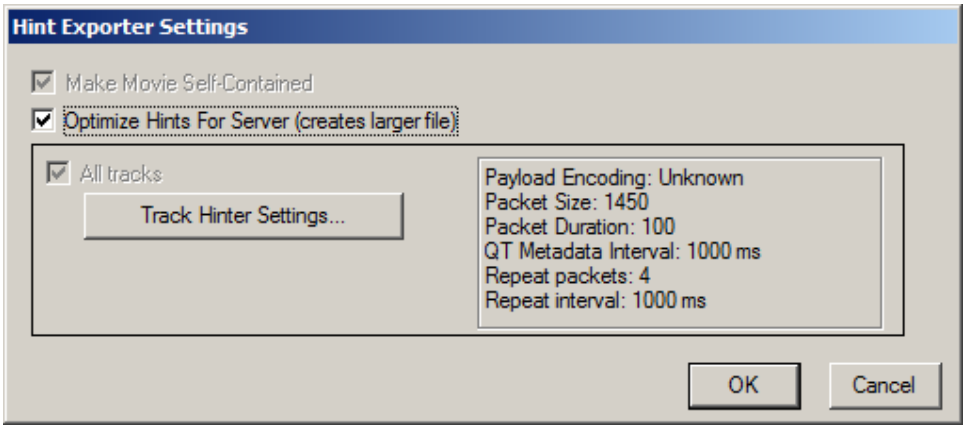


Figure E.6: The Hint Exporter Settings for the video file

APPENDIX F: Making a reference movie to deliver a video on the web

1. Download Apple *MakeRefMovie* utility.
2. Open *MakeRefMovie*, a "*save as*" pop up window.
3. Enter the reference file name of your choice with a .mov file extension.

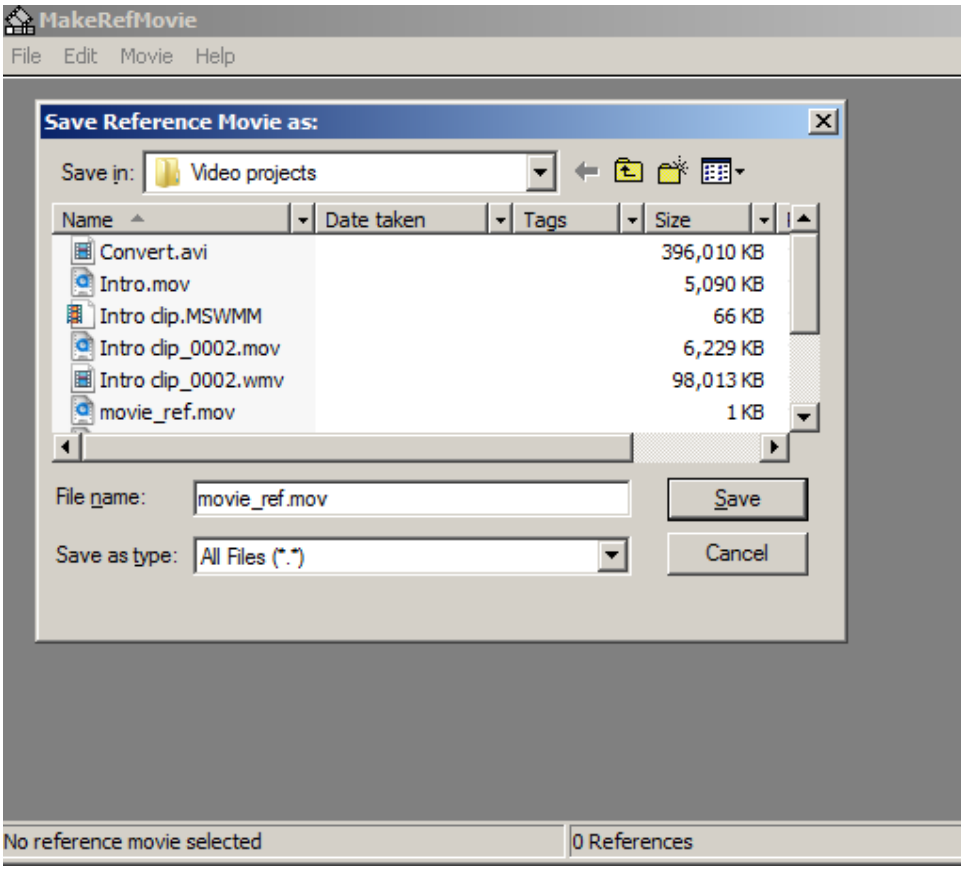


Figure F.1: Creating a new reference movie

4. Click *Movie* from the menu and then click *Add URL*.

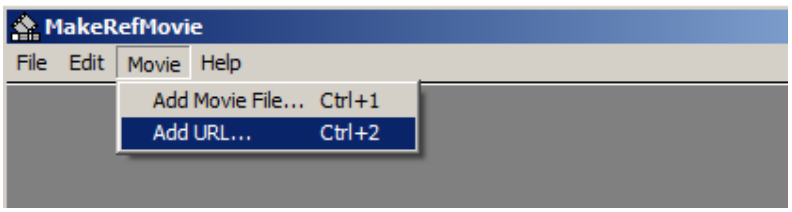


Figure F.2: Adding a URL to the created reference movie

5. Enter the RTSP URL that refers to the file you want to stream.

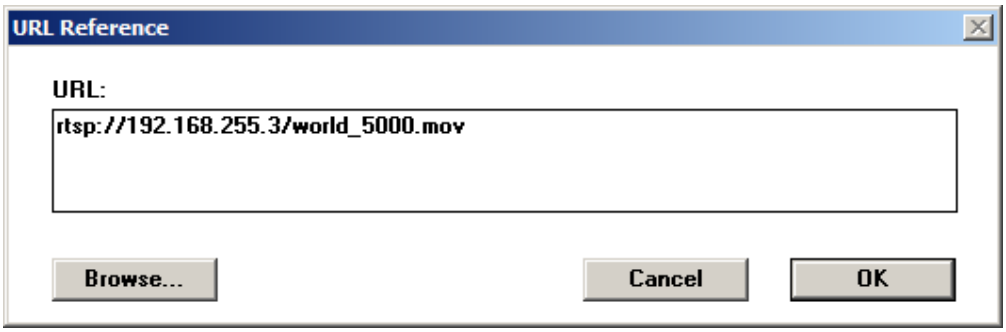


Figure F.3: The RTSP URL for the streaming movie

6. Enter the information as shown in the Figure.

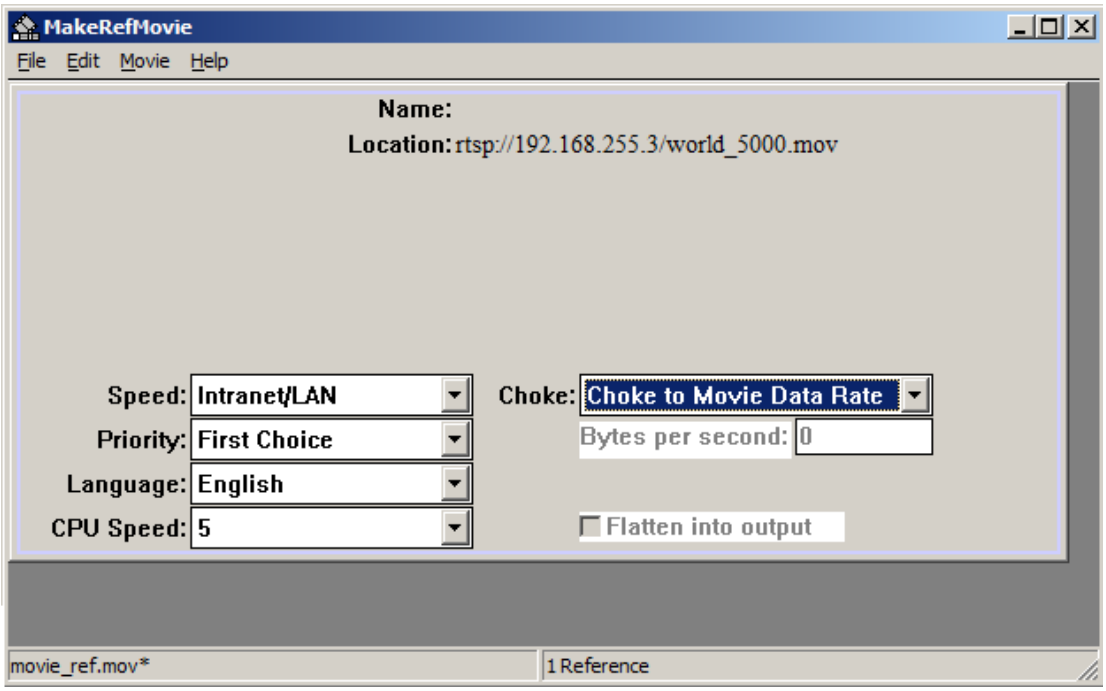


Figure F.4: Specifications for the reference movie

7. Click *File* and then *Save*.